TPC Benchmark™ D (Decision Support Systems) 100 GB Full Disclosure Report

NCR System 4400
Using Teradata RDBMS V2R3.0

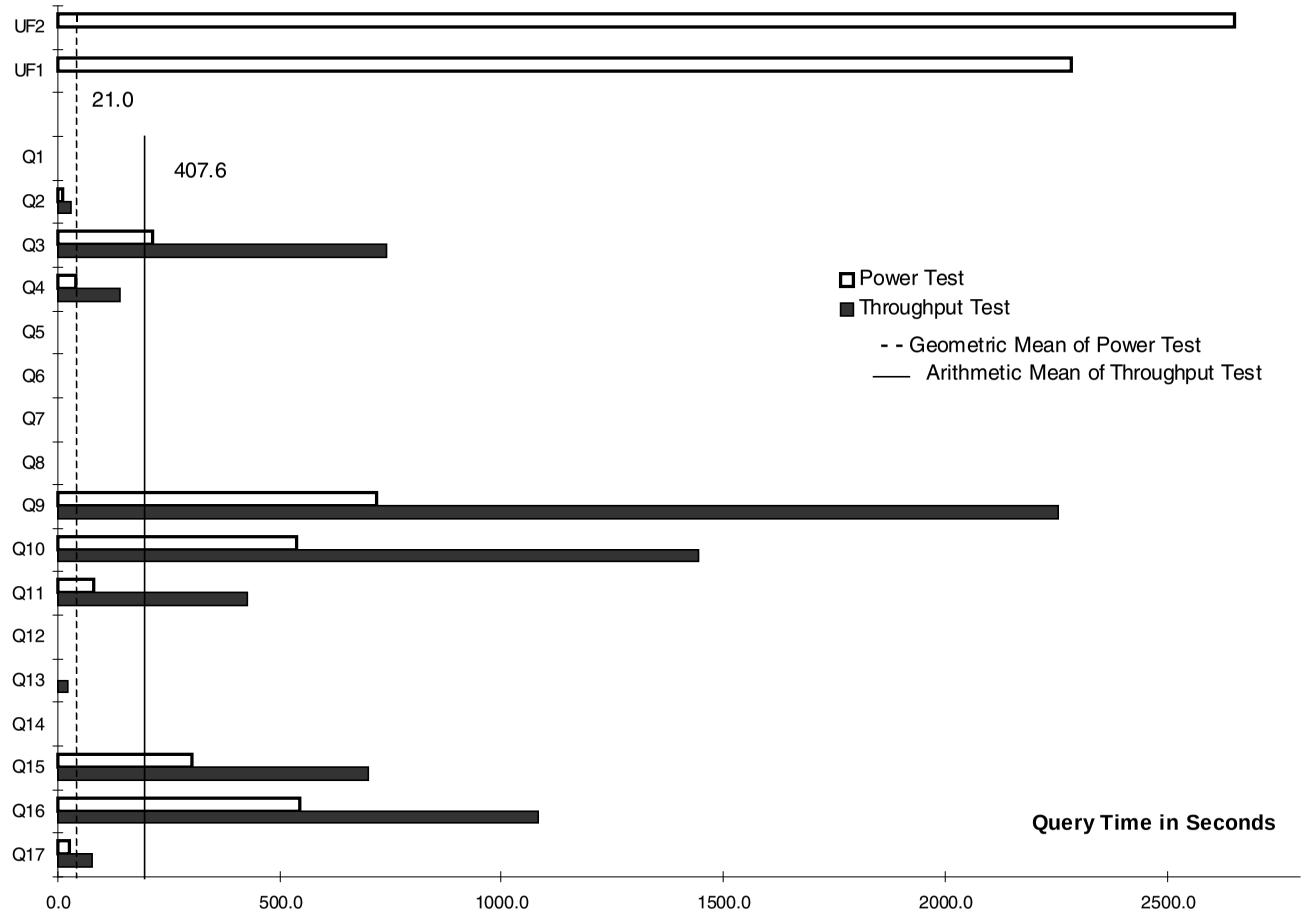
Submitted for Review
June 18, 1999

NCR makes no warranty of any kind with regard to the information contained in this publication, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. NCR shall not be held liable for errors contained herein nor for any damages including loss of profits or other incidental or consequential damages in connection with the furnishing or use of this document.

NCR believes the information contained herein is correct and complete as of the date of publication. The pricing information is believed to accurately reflect the prices in effect for the components, products, and services listed, as of the date of publication, using the pricing methods as described herein. Actual prices to any particular customer are dependent on business agreements, legal contracts, sales and delivery channel arrangements, etc. The performance information is believed to accurately reflect the performance of the components, products, and services listed, as of the date of publication. Actual performance experienced by any particular customer will not necessarily be the same as claimed herein due to the myriad of technical variables such as system layout and configuration, hardware and/or software revision levels and/or change notices, operations parameterization, and background system activity. The content of this document is for informational purposes only.

.

NCR	WorldMark 4400 Using Teradata V2R3.0		TPC-D Rev 1.3.1 Report Date: June 18, 1999	
Total System Cost	TPC-D Power	TPC-D Throughput	Price/Performance	
$321,812	**17,115.2** **QppD@100GB**	**869.1** **QthD@100GB**	**$83** **per QphD@100GB**	
Database Size	Database Manager	Operating System	Other Software	Availability Date
100GB	**Teradata V2R3.0**	**UNIX SVR4** **MP-RAS 03.02.00**	**None**	**HW: 4-16-99** **SW: 8-10-99**



Database Load Time = 47 Hrs 9 Mins	**Total Data Storage / Database Size = 5.34**	**RAID = N**

1 4400 Node:

4 Intel Pentium II Xeon 450 MHz CPUs 2 MB Cache
2 GB Memory
2 Ultra SCSI Adapters
1 CD-ROM Unit, Floppy Drive & Tape Drive
1 Network card
3 9GB internal drives
6 Disk modules, each with
10 x 9 GB drives

Total GB of Storage =534.2

WorldMark 4400
Teradata DBS V2R3.0

TPC-D Rev. 1.3.1

Report Date:
June 18, 1999

Product I.D.	Description	Qty	UNIT PRICE	EXT PRICE	5-YR MAINT
4400-4361-8090	WM4400D/SDW/2PIIX/2M,1GB,27GB	1	$ 39,520	$ 39,520	$ 17,373
4400-F027	Processor, PII Xeon 450/2M w Heatsink	2	$ 8,560	$ 17,120	$ 1,380
4400-F060	Cord, Power - US 120 V	4	$ 40	$ 160	$ -
3498-2244-8090	Monitor 14" Color, MPR II, DDC	1	$ 240	$ 240	$ 226
4400-F048	VRM, Board	3	$ 20	$ 60	$ -
4400-F700	Keyboard - US 120V	1	$ 40	$ 40	$ -
4400-F147	Adpt - PCI Ethernet 10/10	1	$ 76	$ 76	$ -
4400-F118	Memory DIMM 1 GB EDO (4 x 256MB)	1	$ 5,800	$ 5,800	$ -
4400-F129	Adapt - PCI - Quad	3	$ 2,236	$ 6,708	$ -
	Server Hardware			**$ 69,724**	**$ 18,979**
6282-1101-XXXX	Entry level modular JBOD	6	$ 2,250	$ 13,500	$ 25,105
6282-F129	EL 9.1 (1) 10000 RPM Disk Drive	54	$ 1,125	$ 60,750	$ -
9100-K931	Symbios Cable Kit	6	$ 1,688	$ 10,125	$ -
	Storage Devices			**$ 84,375**	**$ 25,105**
F784-1406-0000	V2R3 4400 Up to 4 CPU	1	$ 40,000	$ 40,000	$ 67,500
F481-6030-0000	NFS OE 1-8 users	1	$ 1,520	$ 1,520	$ -
F784-5020-0000	Teradata CLI for SMP Node	1	$ 1,600	$ 1,600	$ 2,430
F784-5021-0000	Teradata BTEQ for SMP Node	1	$ 200	$ 200	$ 304
F784-5022-0000	Teradata FastLoad for SMP Node	1	$ 4,000	$ 4,000	$ 6,075
	Software			**$ 47,320**	**$ 76,309**
	TOTAL			**$ 201,419**	**$ 120,393**
	FIVE YEAR COST OF OWNERSHIP				**$ 321,812**

"Prices used in TPC benchmarks reflect the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices based onassumptions about past or future purchases are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details, see the pricing sections of the TPC benchmark specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org. Thank you."

Audited by: Francois Raab
Information Paradigm

Numerical Quantity Summary

Measurement Results:

Scale Factor	=	100
Total Data Storage / Database Size	=	5.34
Database Load Time	=	47 Hrs 9 Min
Query Streams for Throughput Test	=	3
Geometric Mean of Power Test	=	21.0
TPC-D Power Metric (QppD@100GB)	=	17,115.2
TPC-D Throughput Metric (QthD@100GB)	=	869.1
Composite QphD@100GB	=	3,856.8
Total System Price Over 5 Years	=	$ 321,812
TPC-D Price/Performance Metric	=	$83.44

Measurement Intervals:

Measurement Interval in Throughput Test (Ts) = 21,125 seconds

Duration of stream execution:

Stream ID	Seed	Start-Date	Start-Time	End-Date	End-Time	Total Time
Stream00	68870749	2/5/99	15:56:31	2/5/99	18:00:08	7,417
Stream01	68870750	2/5/99	18:00:08	2/5/99	19:51:08	6,660
Stream02	68870751	2/5/99	18:00:08	2/5/99	20:01:55	7,307
Stream03	68870752	2/5/99	18:00:08	2/5/99	19:53:57	6,829
Updates		2/5/99	18:00:08	2/5/99	23:52:13	21,125

TPC-D Timing Intervals (in seconds):

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Stream0	0.4	10.5	214.5	40.4	0.5	0.2	0.7	0.7	718.6	538.0
Stream01	0.4	51.0	597.8	40.5	0.6	0.4	0.8	4.2	2254.9	1234.1
Stream02	0.8	10.2	681.6	254.5	0.5	1.2	0.8	0.8	2383.0	1658.3
Stream03	0.3	22.0	941.5	126.8	0.5	0.2	2.4	0.9	2129.1	1442.7
Minimum	0.3	10.2	597.8	40.5	0.5	0.2	0.8	0.8	2129.1	1234.1
Average	0.5	27.7	740.3	140.6	0.5	0.6	1.3	2.0	2255.7	1445.1
Maximum	0.8	51.0	941.5	254.5	0.6	1.2	2.4	4.2	2383.0	1658.3

	Q11	Q12	Q13	Q14	Q15	Q16	Q17	UF1	UF2
Stream0	80.8	0.3	2.2	0.2	302.8	543.9	26.6	2282.6	2653.0
Stream01	444.1	0.2	4.4	0.6	666.6	1320.8	34.9	9251.8	2694.2
Stream02	680.2	0.4	54.1	0.4	855.4	567.6	153.4	1914.6	2702.1
Stream03	157.1	0.3	5.2	0.3	582.1	1365.4	49.0	1842.6	2720.3
Minimum	157.1	0.2	4.4	0.3	582.1	567.6	34.9	1842.6	2694.2
Average	427.1	0.3	21.2	0.4	701.4	1084.6	79.1	4336.3	2705.5
Maximum	680.2	0.4	54.1	0.6	855.4	1365.4	153.4	9251.8	2720.3

Information Paradigm

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL

Certified Auditor

Test Sponsors: Alain Crolotte
NCR
100 N. Sepulveda Blvd.
El Segundo, CA 90245

February 9, 1999
I verified the TPC Benchmark™ D performance of the following configuration:
Platform: NCR 4400 Single Node System
DataBase Manager: Teradata V2R3.0
Operating System: UNIX SVR4 MP-RAS 3.02.00
The results were:

CPU's Speed	Memory	Disks	QppD@100GB	QthD@100GB
NCR 4400 Single Node System				
4 Pentium II Xeon (450Mhz)	Cache: 2 MB Main: 2.0 GB	60 x 9.0 GB ext. 3 x 9.0 GB int..	17,115.2	869.1

In my opinion, these performance results were produced in compliance with the TPC requirements for the benchmark. The following verification items were given special attention:

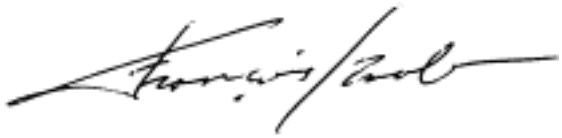
- The TIME table was not used
- The input variables were generated by QGEN
- The database was populated using DBGEN
- The database was maintained using the evolution option
- The throughput metric was computed using the results from three query streams
- The ratio between the longest and the shortest query was such that some adjustments were required
- A compliant implementation specific layer was used
- The query text was produced using compliant minor modifications
- The database records were defined with the proper layout and size
- The database was properly scaled to 100GB and populated accordingly
- The database load time was correctly measured and reported
- The ACID properties were verified and met
- The reported execution times were correctly measured and reported

- Measurement repeatability was verified by two consecutive runs
- At least 8 hours of database log was configured
- The system pricing was verified for major components and maintenance
- The major pages from the FDR were verified for accuracy

Additional Audit Notes:

None.

Respectfully Yours,



François Raab
President

Preface

TPC Benchmark D Overview

The TPC Benchmark™ D (TPC-D) is a Decision Support benchmark. It is a suite of business oriented queries and concurrent updates. The queries and the data populating the database have been chosen to have broad industry-wide relevance while maintaining a sufficient degree of ease of implementation. This benchmark illustrates Decision Support systems that

- *Examine large volumes of data.*
- *Execute queries with a high degree of complexity.*
- *Give answers to critical business questions.*

TPC-D evaluates the performance of various Decision Support systems by the execution of sets of queries against a standard database under controlled conditions. The TPC-D queries:

- *Give answers to real-world business questions.*
- *Are far more complex than most OLTP transactions.*
- *Include a rich breadth of operators and selectivity constraints.*
- *Generate intensive activity on the part of the database server component of the system under test.*
- *Are executed against a database complying to specific population and scaling requirements.*
- *Are implemented with constraints derived from staying closely synchronized with an on-line production database.*

Table of Contents

PREFACE IX

TPC Benchmark D Overview ix

1. GENERAL ITEMS 1

1.1 Benchmark Sponsor(s) 1

1.2 Parameter Settings 1

1.3 Configuration Diagram 2

2. CLAUSE 1 LOGICAL DATABASE DESIGN 3

2.1 Database Definition Statements 3

2.2 Physical Organization 3

2.3 Horizontal Partitioning 3

3. CLAUSE 2 QUERIES AND UPDATE FUNCTIONS 4

3.1 Teradata SQL Used 4

3.2 Verifying Method for Random Number Generation 4

3.3 Generating Values for Substitution Parameters 4

3.4 Query Text and Output Data from Qualification Database 4

3.5 Query Substitution Parameters and Seeds Used 5

3.6 Source Code of Update Functions 5

3.7 Database Maintenance Option 5

4. CLAUSE 3 DATABASE SYSTEM PROPERTIES 6

4.1 Atomicity 6

4.2. Consistency 6

4.3 Isolation 7

4.4 Durability 8

5. CLAUSE 4 SCALING AND DATABASE POPULATION 9

5.1 Ending Cardinality of Tables 9

5.2 Distribution of tables and logs across media 9

5.3 Modifications to the DBGEN 10

5.4 Database Content of Initial Ten Rows 10

5.5 Database Load Time 10

5.6 Database Load Mechanism Details and Illustration 10

5.7 Data Storage Ratio 12

5.7 Data Storage Ratio 12

6. CLAUSE 5 PERFORMANCE METRICS AND EXECUTION RULES 13

6.1 Steps in the Power Test 13

6.2 Timing Intervals for Each Query and Update Functions 14

6.3 Number of Streams for the Throughput Test 14

6.4 Start and End Date/Times for Each Query Stream 14

6.5 Total Elapsed Time of the Measurement Interval 14

6.6 Update Function Start Date/Time and Finish Date/Time 14

6.7 Timing Intervals for Each Query and Each Update Function for Each Stream 15

6.8 Performance Metrics 15

6.9 Reproducibility Method 15

7. CLAUSE 6 SUT AND DRIVER IMPLEMENTATION 16

7.1 Description of Driver Performance Functions 16

7.2 Detailed Description of the Internal Driver Program and Implementation Layer Used 16

8. CLAUSE 7 PRICING 17

8.1 Detailed List of Hardware and Software Used 17

8.2 Total Five Year Price 17

8.3 Committed Delivery Date of General Availability 18

9. CLAUSE 9 AUDIT 19

9.1 Auditor's Information and Attestation Letter 19

APPENDIX A: OPERATING SYSTEMS AND DATABASE PARAMETERS AND OPTIONS 20

A.1 Operating System Parameters 20

A.2 Database Parameters 24

APPENDIX B. DATABASE DEFINITION STATEMENTS 25

B.1 Create Table Statements 25

B.2 Create Index and Collect Statistics Statements 26

APPENDIX C. QUERY TEXT AND OUTPUT DATA FROM QUALIFICATION DATABASE 29

APPENDIX D. QUERY SUBSTITUTION PARAMETERS AND SEEDS USED 39

APPENDIX E. DATABASE CONTENT OF INITIAL TEN ROWS 40

APPENDIX F. DATA LOAD COMPONENTS 44

Description of the loading method 44

F.1 Inmod for FastLoad used by Pfast 45

F.2 Source Code for Pipeline-to-FastLoad Approach 62

F.3 Load Scripts 76

APPENDIX G. DRIVER AND IMPLEMENTATION SPECIFIC LAYER DETAIL 83

Driver program 83

Implementation specific layer 123

1. General Items

1.1 Benchmark Sponsor(s)

A statement identifying the benchmark sponsor(s) and other participating companies must be provided.

NCR is the sponsor of this TPC-D benchmark.

1.2 Parameter Settings

Settings must be provided for all customer-tunable parameters and options which have been changed from the defaults found in actual products, including but not limited to:

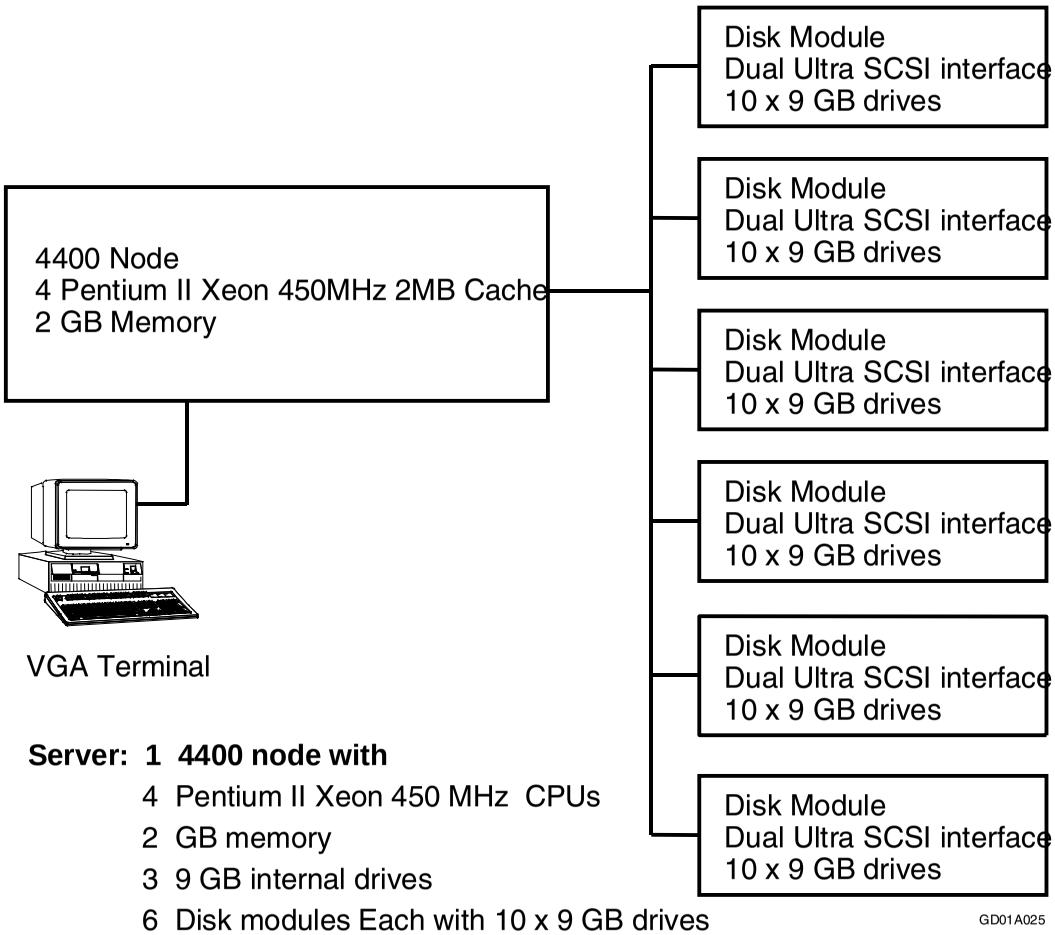
- *Database tuning options*
- *Optimizer/Query execution options*
- *Query processing tool/language configuration parameters*
- *Recovery/commit options*
- *Consistency/locking options*
- *Operating system and configuration parameters*

This requirement can be satisfied by providing a full list of all parameters and options.

See Appendix A for the NCR UNIX SVR4 MP-RAS operating system parameters used to generate the kernel for the configuration used in this benchmark. Also included are all Teradata Version 2 parameters.

1.3 Configuration Diagram

1.3.1 Measured and Priced Configuration Diagram



Server: 1 4400 node with

- 4 Pentium II Xeon 450 MHz CPUs
- 2 GB memory
- 3 9 GB internal drives
- 6 Disk modules Each with 10 x 9 GB drives
- 2 Ultra SCSI Adapters
- 1 Network Card
- 1 CD Rom
- 1 Floppy
- 1 Tape Drive

2. Clause 1 Logical Database Design

2.1 Database Definition Statements

Listings must be provided for all table definition statements and all other statements used to set-up the test and qualification databases.

See Appendix B for the programs and scripts that create tables, build indexes, and define collect statistics, which create and populate the Teradata database for the 4400 TPC-D benchmark.

2.2 Physical Organization

The physical organization of tables and indices, within the test and qualification databases, must be disclosed.

Teradata systems automatically handle data and index entry placement, without any manual organization or setup activity required. During this benchmark, no intervention or special options to Teradata's natural data placement were utilized.

2.3 Horizontal Partitioning

Horizontal partitioning of tables and rows in the test and qualification databases (see Clause 1.5.4) must be disclosed.

While there are some restrictions placed upon physical replication of objects in the test and qualification databases (see Clause 1.5.6), any such replication must be disclosed.

Data is automatically partitioned across all disks evenly, based on a hash value derived from one of the columns of the table. For this benchmark, the first column of every table, with the exception of NATION, was used for hashing. This is the Teradata system default. In the case of NATION, N_NAME was the column selected for hashing.

3. Clause 2 Queries and Update Functions

3.1 Teradata SQL Used

The query language used to implement the queries must be identified.

Teradata SQL was the query language used to implement all queries.

3.2 Verifying Method for Random Number Generation

The method of verification for the random number generation must be described unless the supplied DBGEN and QGEN were used.

The supplied DBGEN and QGEN methods were used to verify random numbers for the 4400 TPC-D benchmark.

3.3 Generating Values for Substitution Parameters

The method used to generate values for substitution parameters must be disclosed. If QGEN is not used for this purpose, then the source code of any non-commercial tool used must be disclosed.

Values for the substitution parameter were generated from a customized internal driver program that embeds the functionality of the QGEN utility. The source code for this program is disclosed in Appendix G.

QGEN Version 1.3.1 was used in this benchmark execution.

3.4 Query Text and Output Data from Qualification Database

The executable query text used for query validation must be disclosed along with the corresponding output data generated during the execution of the query text against the qualification database. The output data for the power and throughput tests must be made available electronically upon request.

Appendix C contains the query text and query output.

3.5 Query Substitution Parameters and Seeds Used

The query substitution parameters used for all performance tests must be disclosed in tabular format, along with the seeds used to generate these parameters.

Appendix D contains the seed and query substitution parameters.

3.6 Source Code of Update Functions

The details of how the update functions were implemented must be disclosed (including source code of any non-commercial program used).

A specially-written driver program was implemented to execute both the power and the throughput tests, including UF1 and UF2, the update functions. This program uses DBGEN functionality to produce data and primary keys for use in both single and multi-stream tests.

Appendix G contains the source code of this program.

3.7 Database Maintenance Option

The details of the database maintenance option selected (i.e., reset or evolve) must be disclosed (including source code of any non-commercial program used).

The database maintenance option selected was evolve.

4. Clause 3 Database System Properties

4.1 Atomicity

4.1.1 Completed Transaction

Perform the ACID Transaction for a randomly selected set of input data and verify that the appropriate rows have been changes in the ORDER, LINEITEM, and HISTORY tables.

The ORDER table total price and the LINEITEM table extended price were accessed, as well as a row count for the HISTORY table. The ACID Transaction was performed against ORDER and LINEITEM rows, changes were made and verified to the LINEITEM and ORDER tables, and it was proven that the correct data was inserted into the HISTORY table. Verification was done using these formulas:

- L_Extendedprice = L_Extendedprice + ((delta)*(L_Extendedprice/L_Quantity))
- O_Totalprice = O_Totalprice+(O_Cost*(1-L_Discount)*(1+L_Tax))

4.1.2 Aborted Transaction

Perform the ACID Transaction for a randomly selected set of input data, substituting a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that the appropriate rows have not been changed in the ORDER, LINEITEM, and HISTORY tables.

A count of the number of HISTORY table rows was obtained. The ACID Transaction was performed against randomly-selected rows, substituting a ROLLBACK for the COMMIT of the transaction.

The ORDER total price and the LINEITEM extended price were verified to be the same as they were prior to executing the ACID Transaction. Further, it was verified that no HISTORY record was inserted.

4.2. Consistency

Consistency is the property of the application that requires any execution of transactions to take the database from one consistent state to another.

4.2.1 Consistency Test

Verify that ORDER and LINEITEM tables are initially consistent, submit the prescribed number of ACID Transactions with randomly selected input parameters, and re-verify the consistency of the ORDER and LINEITEM tables involved.

The initial consistency of the ORDER and LINEITEM tables was established, by proving this condition: O_Totalprice = Sum(L_Extendedprice*(1-L_Discount)*(1+L_Tax)). The prescribed number of ACID Transactions were executed and the consistency of the involved ORDER and LINEITEM rows was re-verified.

4.3 Isolation

Operations of concurrent transactions must yield results which are indistinguishable from the results which would be obtained by forcing each transaction to be serially executed to completion in some order.

4.3.1 Read-Write Conflict with Commit

Demonstrate isolation for the read-write conflict of a read-write transaction and a read-only transaction when the read-write transaction is committed.

Following the guidelines in the TPC-D Specification document, clause 3.4.2.1, it was verified that the second transaction does not see the first transaction's updates, and does not complete, until the first transaction has committed.

4.3.2 Read-Write Conflict with Rollback

Demonstrate isolation for the read-write conflict of a read-write transaction and a read-only transaction when the read-write transaction is rolled back.

Following the guidelines in the TPC-D Specification document, clause 3.4.2.2, it was verified that the second transaction does not see the first transaction's updates, and that the second transaction does not complete until the first transaction has rolled back.

4.3.3 Write-Write Conflict with Commit

Demonstrate isolation for the write-write conflict of two update transactions when the first transaction is committed.

Following the guidelines in the TPC-D Specification document, clause 3.4.2.3, it was verified that the second transaction, which attempts to update the same data as the first transaction, waits until the first transaction completes, and receives data that reflects the changes made by the first transaction.

4.3.4 Write-Write Conflict with Rollback

Demonstrate isolation for the write-write conflict of two update transactions when the first transaction is rolled back.

Following the guidelines in the TPC-D Specification document, clause 3.4.2.4, it was verified that the second transaction, which attempts to update the same data as the first transaction, waits until the first transaction rollbacks, and receives data that reflects the original state of the data, not the changes made by the first transaction.

4.3.5 Read and Write Against Different Tables Concurrently

Demonstrate the ability of read and write transactions affecting different database tables to make progress concurrently.

Based on the TPC-D Specification document, clause 3.4.2.5, it was proven that a second transaction that updates tables in the database not referenced by an earlier-started read-only transaction will complete prior to the read-only transaction. Verification was made that the appropriate rows were updated and that the update transaction started after and completed before the read only transaction.

4.3.6 Updates Not Indefinitely Delayed by Reads on Same Table

Demonstrate that the continuous submission of arbitrary (read-only) queries against one or more tables of the database does not indefinitely delay update transactions affecting those tables from making progress.

Following TPC-D Specification document, clause 3.4.2.6, process 1 submitted Query 1 (a long-running query) for execution, followed by a second process that executed a short update transaction against the same table. A third process submitted a second long-running query against the same table while the original process 1 was still executing. Process 1's query completed first, followed by process 2's update transaction, and lastly process 3's query completed. It was verified that the correct rows were updated and that updates are executed in a timely manner.

4.4 Durability

The tested system must guarantee durability: the ability to preserve the effects of committed transactions and insure database consistency after recovery from any one of the failures listed in Clause 3.5.3.

4.4.1 Failure of a Durable Medium

Guarantee the database and committed updates are preserved across a permanent irrecoverable failure of any single durable medium containing TPC-D database tables or recovery log data.

A disk failure was caused in the midst of the required durability test, the system was restarted, and the durability success file and the HISTORY table were compared successfully. The process recorded in the TPC-D Specification Document, Clause 3.5.2 was adhered to during this test.

4.4.2 System Crash

Guarantee the database and committed updates are preserved across an instantaneous interruption (system crash / system hang) in processing whcih requires the system to re-boot to recover.

A system interruption was caused in the midst of the required durability test, the system was restarted, and the durability success file and the HISTORY table were compared successfully. The process recorded in the TPC-D Specification Document, Clause 3.5.3 as adhered to during this test.

4.4.3 Memory Failure

Guarantee the database and committed updates are preserved across failure of all or part of memory (loss of contents).

A memory failure was caused in the midst of the required durability test, the system was restarted, and the durability success file and the HISTORY table were compared successfully. The process recorded in the TPC-D Specification Document, Clause 3.5.4 was adhered to during this test.

5. Clause 4 Scaling and Database Population

5.1 Ending Cardinality of Tables

The cardinality (e.g., the number of rows) of each table of the test database, as it existed at the completion of the database load (see Clause 4.2.5), must be disclosed.

Order	150,000,000
Lineitem	600,037,902
Customer	15,000,000
Part	2,000,000
Supplier	1,000,000
Partsupp	80,000,000
Nation	25
Region	5

5.2 Distribution of tables and logs across media

The distribution of tables and logs across all media must be explicitly described.

In this configuration there are 6 JBOD units. Each unit contains 10 disks for a total of 60.

The 4400 node has been configured with 20 virtual AMPs.

The Teradata database system enforces the following data placement rules:

- The rows of each table are randomly assigned to one of the 20 virtual AMPs.
- Each virtual AMP controls approximately an equal portion of each tables' rows.
- There are no partitioning or data placement options available.
- There is no knowledge of which physical disk specific rows reside on.
- The transient journal, responsible for logging before images of data changes, is local to each virtual AMP, and is therefore dispersed across disks similar to user data.

Because there is no partitioning scheme, and data is evenly dispersed across all the vprocs, the following distribution summary is offered.

The 4400 node is associated with 60 disks within 6 JBOD units.

The following is a description of data distribution across internal disks and external disk units:

Component	Description of Content
Internal 1	Operating system, root
Internal 2	System page and swap
JBOD Unit 1	16.66% of LINEITEM, ORDER, CUSTOMER, PART, SUPPLIER, PARTSUPP, NATION, REGION, TRANSIENT JOURNAL
JBOD Unit 2	16.66% of LINEITEM, ORDER, CUSTOMER, PART, SUPPLIER, PARTSUPP, NATION, REGION, TRANSIENT JOURNAL
JBOD Unit 3	16.66% of LINEITEM, ORDER, CUSTOMER, PART, SUPPLIER, PARTSUPP, NATION, REGION, TRANSIENT JOURNAL
JBOD Unit 4	16.66% of LINEITEM, ORDER, CUSTOMER, PART, SUPPLIER, PARTSUPP, NATION, REGION, TRANSIENT JOURNAL
JBOD Unit 5	16.66% of LINEITEM, ORDER, CUSTOMER, PART, SUPPLIER, PARTSUPP, NATION, REGION, TRANSIENT JOURNAL
JBOD Unit 6	16.66% of LINEITEM, ORDER, CUSTOMER, PART, SUPPLIER, PARTSUPP, NATION, REGION, TRANSIENT JOURNAL

NOTE: As the Region and Nation table have only 5 and 25 rows respectively, some virtual AMPs will not be assigned rows from those smaller tables.

5.3 Modifications to the DBGEN

Any modifications to the DBGEN (see Clause 4.2.1) source code (provided in Appendix B) must be disclosed. In the event that a program other than DBGEN was used to populate the database, it must be disclosed in its entirety.

Appendix F contains the source of the inmods for the PFast utility program. Section 5.6 of this FDR provides detail about this utility program.

5.4 Database Content of Initial Ten Rows

The content of the first 10 rows of each table in the test database must be disclosed.

Appendix E contains the first 10 rows of each table in the test database.

5.5 Database Load Time

The database load time for the test database (see Clause 4.3) must be disclosed.

The database load time was 47 hours 9 minutes.

5.6 Database Load Mechanism Details and Illustration

The details of the database load mechanism must be described and illustrated with a block diagram.

FastLoad utility was used and is described and illustrated below.

5.6.1 Parallel FastLoad

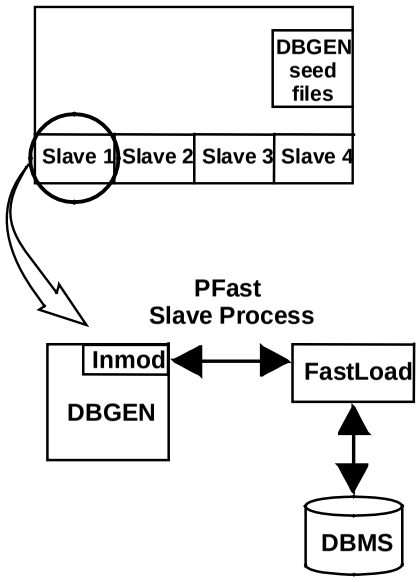
Parallel FastLoad (PFast) is a product that extends the functionality of the standard Teradata FastLoad utility. PFast, which extends parallelism to the client portion of FastLoad, has two components: A master process and some number of slave processes. The master process begins and coordinates the entire database load activity, while each of the slave processes do the actual generation (or reading) or rows and FastLoad execution. PFast loads data to a table through multiple parallel streams.

For this benchmark, PFast is used to load all tables. A total of 4 slave processes are activated, except for Nation and Region tables which used 1 slave.

Each slave process is responsible for generating and loading 1/4 of each table's rows. Using DBGEN-generated seed files the slave process, based on his assigned Child-ID, chooses which seeds are appropriate for his 1/4 of the data.

The FastLoad utility is the driver within each slave process, calling DBGEN as additional rows are required. The generated rows then pass through an inmod (specific to the TPC-D database), and control is passed back to the FastLoad instance who loads the data. Each slave has a copy of the DBGEN code and the inmod.

Appendix F contains the inmods used by Pfast.



GD01A012

5.7 Data Storage Ratio

The data storage ratio must be disclosed. It is computed as the ratio between the total amount of priced disk space, and the chosen test database size as defined in Clause 4.1.3.

The data storage ratio is computed based on the following information:

5.7 Data Storage Ratio

The data storage ratio must be disclosed. It is computed as the ratio between the total amount of priced disk space, and the chosen test database size as defined in Clause 4.1.3.

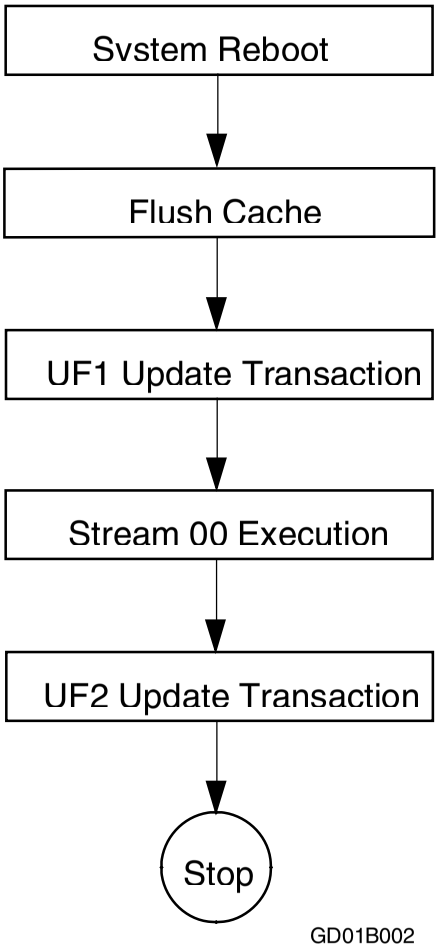
The data storage ratio is computed based on the following information:

Disk Type	# of Disks	Space per Disk	Sub-Total Disk Space	Total Size	Data Storage Ratio
External	60	8.48 GB	508.8 GB		
Internal	3	8.48 GB	25.44 GB	534.24 GB	5.34

6. Clause 5 Performance Metrics and Execution Rules

6.1 Steps in the Power Test

The details of the steps followed to implement the power test (e.g., system boot, database restart, etc.) must be disclosed.



6.2 Timing Intervals for Each Query and Update Functions

The timing intervals for each query of the measured set and for both update functions must be reported for the power test.

The Power Test timing intervals can be found in the Executive Summary which is placed at the beginning of this document, under the Numerical Quantity Summary.

6.3 Number of Streams for the Throughput Test

The number of execution streams used for the Throughput Test must be disclosed.

The Throughput Test used 3 execution streams.

6.4 Start and End Date/Times for Each Query Stream

The start time and finish time for each query execution stream must be reported for the throughput test.

The Throughput Test start time and finish time for each stream is reported in the Executive Summary of this document within the Numerical Quantity Summary.

6.5 Total Elapsed Time of the Measurement Interval

The total elapsed time of the measurement interval.

The total elapsed time of the throughput test was 21,125 seconds.

6.6 Update Function Start Date/Time and Finish Date/Time

Start and finish time for each update function in the update stream must be reported for the throughput test.

Stream / Function	Start-Date	Start-Time	End-Date	End-Time
Stream1-UF1	02/05/99	20:01:55	02/05/99	20:34:20
Stream1-UF2	02/05/99	20:34:20	02/05/99	21:19:14
Stream2-UF1	02/05/99	21:19:14	02/05/99	21:51:09
Stream2-UF2	02/05/99	21:51:09	02/05/99	22:36:11
Stream3-UF1	02/05/99	22:36:11	02/05/99	23:06:53
Stream3-UF2	02/05/99	23:06:53	02/05/99	23:52:13

6.7 Timing Intervals for Each Query and Each Update Function for Each Stream

The timing intervals for each query of each stream and for each update function must be reported for the Throughput Test.

The timing intevals for each query and update function for the Throughput Test is reported in the Executive Summary at the beginning of this document, within the Numerical Quantities Summary.

6.8 Performance Metrics

Verify that the metrics are computed as required.

The performance metrics, and the numbers on which they are based, are contained within the Executive Summary at the beginning of this document, within the Numerical Quantities Summary.

6.9 Reproducibility Method

A description of the method used to determine the reproducibility of the measurement results must be reported. This must include the performance metrics (QppD, QthD and QphD) from the reproducibility runs.

Two consecutive runs were executed. The following table contains the reproducibility metrics for the system reported in this document.

System	QppD@100GB	QthD@100GB	QphD@100GB
Run 1	17,115.2	869.1	3,856.8
Run 2	17,049.5	878.3	3,869.7

7. Clause 6 SUT and Driver Implementation

7.1 Description of Driver Performance Functions

A detailed description of how the driver performs its functions must be supplied, including any related source code or scripts. This description should allow an independent reconstruction of the driver.

No driver external to the SUT was used.

7.2 Detailed Description of the Internal Driver Program and Implementation Layer Used

If an implementation specific layer is used, then a detailed description of how it performs its functions must be supplied, including any related source code or scripts. This description should allow an independent reconstruction of the implementation specific layer.

The internal driver program used in this benchmark utilizes the standard Teradata call-level interface (CLIv2). An implementation specific layer was added between the driver program and CLIv2 to hide some of the complexities of CLIv2 from the main body of the program. This program provides the following functionality:

- The program reads the same input files as QGEN and generates the query text with the appropriate input variables.
- The program directly submits the queries, with a time-date stamp between each query submission .
- The program produces two output files: 1) A summary output that includes the query times, and 2) Detail output which includes all of the query text and output.

Source code of the Driver Program and Implementation Specific Layer is included in Appendix G.

8. Clause 7 Pricing

8.1 Detailed List of Hardware and Software Used

A detailed list of hardware and software used in the priced system must be reported. Each item must have vendor part number, description, and release/revision level, and either general availability status or committed delivery date. If package-pricing is used, contents of the package must be disclosed. Pricing source(s) and effective date(s) of price(s) must also be reported.

A detailed list of hardware and software used in this benchmark, as well as the total five year price calculations, is listed in the Executive Summary at the beginning of this document. This pricing information was verified with the NCR Customer Support Services in Dayton, Ohio.

8.2 Total Five Year Price

The total 5-year price of the entire configuration must be reported, including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.

NCR's Standard Dollar Volume Discount has been used in the pricing of this configuration . This discount is based on various levels of Dollar Volume. Enterprise System Support is NCR's new warranty and post warranty support structure and is the standard maintenance structure for NCR WorldMark Systems. ESS consists of three levels of service--Standard, Enhanced, and Premium which are structured to meet increasingly critical business operating environments.

NCR items carry a one-year warranty except for the clients, UNIX and Teradata. Four additional years of maintenance have been priced to meet TPC requirements for five-year maintenance. NCR's Standard Support tier with upgraded hours of coverage (7x24) and response times (4 hours) has been used for NCR's TPC-D reporting. On-site repair averages four hours for CPU modules and four hours for all other failures. Response time to service calls is priced to meet TPC-D benchmark Standard Specification requirements. Software maintenance includes all upgrades to the operating system and minor point releases for the Teradata DBS for UNIX. Any items on the pricing sheet with NCR part numbers and having no maintenance rate listed have their maintenance cost included in the system maintenance rate.

The Fastload utility is bundled into the price of the Teradata Client software.

PFast is a free, general purpose parallel utility which uses the standard Teradata call-level interface (CLIv2). This utility is available to the public and can be obtained from the following location on the internet: http://www.ncr.com/product/teradata/.

8.3 Committed Delivery Date of General Availability

The committed delivery date for general avilability of products used in the price calculations must be reported. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available.

Teradata V2R3.0 software used in this benchmark will be available for customer use on August 10, 1999.

4400 hardware is available now.

9. Clause 9 Audit

9.1 Auditor's Information and Attestation Letter

The auditor's agency name, address, phone number, and Attestation letter with a brief audit summary report indicating compliance must be included in the full disclosure report. A statement should be included specifying who to contact in order to obtain further information regarding the audit process.

An attestation letter is attached to the summary report, as well as the auditor's name and address. This letter was submitted at the time of benchmark publication.

Appendix A: Operating Systems and Database Parameters and Options

A.1 Operating System Parameters

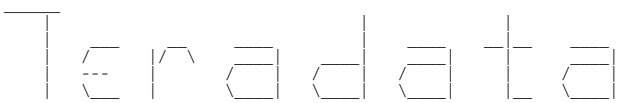
```
****************************************

   Unix mtune file

****************************************

* Lines ending in "%%INS%%" are from mtune.d/*       %%INS%%
* and constructed automatically.             %%INS%%
* DO NOT edit manually.                      %%INS%%
YNET    1       1       1       %%INS%%
* The following lines, if any, are entries   %%INS%%
* preserved from the previous mtune file.    %%INS%%
* Copyright 1994. 1996 AT&T Global Information Solutions - Dayton,
Ohio, USA
*
*       Copyright (c) 1990 UNIX System Laboratories, Inc.
*       Copyright (c) 1984, 1986, 1987, 1988, 1989, 1990 AT&T
*         All Rights Reserved
*
*       THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF
*       UNIX System Laboratories, Inc.
*       The copyright notice above does not evidence any
*       actual or intended publication of such source code.
*
*       Copyright (c) 1987, 1988 Microsoft Corporation
*         All Rights Reserved
*
*       This Module contains Proprietary Information of Microsoft
*       Corporation and should be treated as Confidential.
*
*ident  "@(#):mtune     23.1.2.3"
*
* Modifications & additions have been made to the mtune file for
performance
* tuning.  Consult tuning documents for explanations.
*     Modified:  BUFHWM    SYSSEGSZ  SEGMAPSZ  GPGSLO  GPGSHI
*        Added:  LOTSFREE  DESFREE   MINFREE
*      Deleted:  SYSSEGSZ
*
* "OBSOLETE" in second field means parameter is no longer used by the
OS.
*    The parameter may be removed in a later version of the OS.
Idconfig
*    will a print warning if the parameter is used in stune.
*
* General Kernel Parameters ------------------
TRACESZ         8192    4096    8192
NCPU            32      1       32
NCALL           60      30      700
NFILE           OBSOLETE
NMOUNT          OBSOLETE
NPROC           250     50      10240
NREGION         OBSOLETE
NCLIST          120     1       400
MAXUP           30      15      2048
NOFILES         OBSOLETE
NHBUF           64      32      1024
NPBUF           20      20      8192
NAUTOUP         60      0       120
FDFLUSHR        1       1       1
BDFLUSHR        OBSOLETE
MAXPMEM         0       0       16384
SHLBMAX         3       2       6
FLCKREC         300     100     2000
PUTBUFSZ        5120    2000    10000
MAXSLICE        100     25      100
ULIMIT          OBSOLETE
SPTMAP          400     50      600
PIOMAP          50      50      50
PIOMAXSZ        64      4       64
MAXMINOR        0x3fffff        255     0x3fffff
NGROUPS_MAX     16      1       16
NBUF            100     100     3000
BUFHWM          0       0       65536
ARG_MAX         5120    1024    51200
RSTCHOWN        0       0       1
MAXLINK         2048    1000    32767
* File System Parameters ---------------
NINODE          500     100     5000
NS5INODE        OBSOLETE
UFSNINODE       500     100     5000
VX_NINODE        0       0       10000
NDQUOT          200     100     400
NRNODE          300     100     1300
S52KNBUF        OBSOLETE
S52KNHBUF       OBSOLETE
* ------ CDFS Parameters -----
NCDINODE        256     64      2048
NCDEXTENT       OBSOLETE
NCDFILSYS       OBSOLETE
* Paging Parameters -----------------
VHNDFRAC        OBSOLETE
AGEINTERVAL     OBSOLETE
```

```
MINPAGEFREE     64      64      64
GPGSLO          25      0       25
GPGSHI          OBSOLETE
GPGSMSK         OBSOLETE
MAXSC           OBSOLETE
MAXFC           OBSOLETE
MAXUMEM         OBSOLETE
MINARMEM        100     100     160
MINASMEM        25      25      40
MINAKMEM        16      4       64
MINHIDUSTK      4       4       32
MINUSTKGAP      2       2       32
PAGES_UNLOCK    200     200     200
* Pageout Daemon Parameters
LOTSFREE        0       0       512
DESFREE         0       0       256
MINFREE         0       0       128
* STREAMS Parameters ----------------
NQUEUE          OBSOLETE
NSTREAM         OBSOLETE
NSTRPUSH        9       9       10
NSTREVENT       OBSOLETE
MAXSEPGCNT      OBSOLETE
NMUXLINK        OBSOLETE
STRMSGSZ        0       0       32767
STRCTLSZ        1024    1024    1024
STRTHRESH       0x400000        0       0x2000000
NBLK4096        OBSOLETE
NBLK2048        OBSOLETE
NBLK1024        OBSOLETE
NBLK512         OBSOLETE
NBLK256         OBSOLETE
NBLK128         OBSOLETE
NBLK64          OBSOLETE
NBLK16          OBSOLETE
NBLK4           OBSOLETE
STRLOFRAC       OBSOLETE
STRMEDFRAC      OBSOLETE
* --------- muoe952386: Made FASTBUF tunable --------------------------
------
* If the value is set to 0, then the system will determine the default
value
* for this variable based on system size.
*
* NOTE: If FASTBUF value was changed from 0 the new value should be 128
atleast
*       and multiple of four bytes.
* ------------------------------------------------------------------
------
FASTBUF         0       0       512
* ------ log (strlog) related tunable parameters
NLOG            10      3       16
MAXWERRMSG      80      0       100
MAXWTRCMSG      80      0       100
MAXWCONSMSG     80      0       100
NUMSP           64      5       128
NUMTIM          128     1       8192
NUMTRW          16      1       8192
TIM_HIWATER     OBSOLETE
TIM_LOWATER     OBSOLETE
TRW_HIWATER     65535   512     1048576
TRW_LOWATER     40960   512     1048576
SOCK_HIWATER    8192    512     1048576
SOCK_LOWATER    4096    128     1048576
NUMSAD          8       1       16
NSTRPHASH       64      16      512
NAUTOPUSH       32      32      32
* Message Parameters ---------------
MSGMAP          100     10      400
MSGMAX          2048    512     8192
MSGMNB          4096    4096    65535
MSGMNI          50      50      1000
MSGSSZ          8       8       8
MSGTQL          40      40      512
MSGSEG          1024    1024    65535
* Semaphore Parameters -------------
SEMMAP          25      10      1000
SEMMNI          25      10      4096
SEMMNS          60      60      4096
SEMMNU          30      30      10240
SEMMSL          25      25      1000
SEMOPM          10      10      32
SEMUME          10      10      32
SEMVMX          32767   32767   32767
SEMAEM          16384   16384   16384
* Shared Memory Parameters ----------
SHMMAX          524288  131072  0x7FFFFFFF
SHMMIN          1       1       1
SHMMNI          100     100     500
SHMSEG          6       6       64
SHMALL          OBSOLETE
* Shared Memory Nailing GIDs ---------
SHM_NAILED_GID1         0       0       2147483647
SHM_NAILED_GID2         0       0       2147483647
SHM_NAILED_GID3         0       0       2147483647
SHM_NAILED_GID4         0       0       2147483647
SHM_NAILED_GID5         0       0       2147483647
SHM_NAILED_GID6         0       0       2147483647
SHM_NAILED_GID7         0       0       2147483647
SHM_NAILED_GID8         0       0       2147483647
SHM_NAILED_GID9         0       0       2147483647
* RFS Parameters ---------------------
NRCVD           150     40      500
NSNDD           150     100     350
NSRMOUNT        20      1       50
NADVERTISE      OBSOLETE
MAXGDP          24      10      32
MINSERVE        3       3       3
MAXSERVE        6       3       6
NRDUSER         250     1       700
RFHEAP          OBSOLETE
NLOCAL          OBSOLETE
NREMOTE         OBSOLETE
RCACHETIME      10      -1      10
RFS_VHIGH       OBSOLETE
RFS_VLOW        OBSOLETE
```

```
RF_MAXKMEM      0       0       50
* XENIX Parameters ------------------
DSTFLAG         1       0       1
NSCRN           0       0       10
NEMAP           10      10      10
TIMEZONE        480     0       1440
XSEMMAX         60      0       60
XSDSEGS         25      0       100
XSDSLOTS        3       0       5
* Miscellaneous Parameters ----------------------
DO386B1         2       0       2
DO387CR3        2       0       2
SANITYCLK       0       0       1
DMAEXCL         1       0       1
MAXDMAPAGE      0       0       524288
DMAABLEBUF      70      10      100
KDBSYMSIZE      200000  10000   400000
PIOSEGSZ        1024    1024    1024
SEGMAPSZ        0       0       51200
FORCESMALLMEMORY 0      0       1
* Device Driver Parameters ---------
NUMXT           3       1       3
NUMSXT          6       1       6
NCPYRIGHT       10      10      10
NKDVTTY         OBSOLETE
PRFMAX          4096    2048    10240
CMF             1       0       1
RCMF            0       0       1
COM2CONS        0       0       1
RIDEOUT         OBSOLETE
MNR_ON          OBSOLETE
SANECNT         OBSOLETE
USANEON         OBSOLETE
* ASYNCIO Parameters ----------------
NAIOSYS         300     0       4096
MINAIOS         OBSOLETE
MAXAIOS         5       1       64
AIOTIMEOUT      OBSOLETE
NAIOPROC        OBSOLETE
* EVENTS Parameters----------------------------
MEVQUEUES       50      50      50
MEVKEVS         50      50      50
MEVEXREFS       50      50      50
MEVEXPRS        50      50      50
MEVTERMS        250     250     250
MEVSEXPRS       50      50      50
MEVSTERMS       100     100     100
MEVTIDS         100     100     100
MEVRETRYS       50      50      50
MEVEXITS        50      50      50
MEVSIGS         50      50      50
MEVSTRDS        50      50      50
MEVDIRENTS      50      50      50
EVDATA          20480   20480   20480
EVTIDHTS        128     128     128
EVFNHTS         256     256     256
EVMAXEV         20      20      20
EVMAXDPE        1024    1024    1024
EVMAXMEM        10240   10240   10240
EVMAXTRAPS      25      25      25
EVMAXETERMS     20      20      20
*  Timer and Scheduler Parameters ------------------
HRTIME          50      50      50
HRVTIME         50      50      50
RTMAXPRI        OBSOLETE
RTNPROCS        OBSOLETE
TSMAXUPRI       20      10      30
TSNPROCS        OBSOLETE
MAXCLSYSPRI     160     160     160
*  Affinity Scheduling Tunables --------------------
AFFIN_ON        1       0       1
AFFINDECAY      16      0       512
* Resource Limit Parameters ------------------
*
* Default per process resource limits (set to 0x7FFFFFFF for infinite
limit)
* S prefix is for soft limits, H prefix is for hard limits
*
* CPULIM - maximum combined user and system time in seconds
* FSZLIM - maximum file size in bytes
* DATLIM - maximum writeable mapped memory (swap space) in bytes
* STKLIM - maximum size of current stack in bytes
* CORLIM - maximum size of core file in bytes
* FNOLIM - maximum number of file descriptors
* VMMLIM - maximum amount of simultaneously mapped virtual memory in
bytes
SCPULIM         0x7FFFFFFF      60              0x7FFFFFFF
HCPULIM         0x7FFFFFFF      60              0x7FFFFFFF
SFSZLIM         0x7FFFFFFF      0x100000        0x7FFFFFFF
HFSZLIM         0x7FFFFFFF      0x100000        0x7FFFFFFF
SDATLIM         0x1000000       0x1000000       0x7FFFFFFF
HDATLIM         0x1000000       0x1000000       0x7FFFFFFF
SSTKLIM         0x1000000       0x2000          0x7FFFFFFF
HSTKLIM         0x1000000       0x2000          0x7FFFFFFF
SCORLIM         0x2000000       0x100000        0x7FFFFFFF
HCORLIM         0x2000000       0x100000        0x7FFFFFFF
SFNOLIM         0x40            0x20            0x400
HFNOLIM         0x400           0x20            0x400
SVMMLIM         0x1000000       0x1000000       0x7FFFFFFF
HVMMLIM         0x1000000       0x1000000       0x7FFFFFFF
*  Number of overflow buffers in pageio_setup
PGOVERFLOW      16      8       128
NOTPGOVERFLOW   16      8       128
*  bdevcnt, cdevcnt tunable
BMAX            25      20      256
CMAX            40      30      256
NCR_3500        0       0       0
MULTI_PROC      0       0       0
* ------ Level 5 Tunable Parameters ------
L5SysInt_Type   5       4       5
L5NMIenables    0xff    0       0xff
L5SysIntenables         0       0       0xff
L5CPUenables    0xef    0       0xff
L5McAddrenables         0xfd    0       0xff
L5McDataenables         0x87    0       0xff
L5DMAenables    0xc0    0       0xff
```

```
SBIthreshold   50      0       0x7fffffff
SBIdisablePeriod       1440    0       0x7fffffff
L5WatchDogPeriod       15      0       0x7fffffff
L5WatchDogPFRPeriod    1800    0       0x7fffffff
L5FrontSwitchTime      0       -2      60
* ------ End of Level 5 Tunable Parameters ------
* ------ performance Group Tunables ------
JQF_CLIENTS    2500    100     3000
JQF_SIZE       40      40      300
JQF_SESS       10      10      50
PW_MAX_ORA_PID OBSOLETE
PW_MAX_EVENT   OBSOLETE
* muoe961889 : hbc - 27 Jun 1996
* Increased LISTIO_MAX_CNT to 1024
LISTIO_MAX_CNT 1024    1       2048
TUBESIZE       1024    1024    4096
TUBETYPE       OBSOLETE
* ------ End of Performance group  Tunables ------
* SUM Privilege Module tunable -----------------------
PRIVID  0       0       0
* ------ Override Autocad lockout mechanism ------
ACADOVERRIDE   0       0       1
* tunables for loadable kernel modules
BDEV_RESERVE   10      0       255
CDEV_RESERVE   10      0       255
FMOD_RESERVE   15      0       255
VFS_RESERVE    10      0       255
DEF_UNLOAD_DELAY       600     0       3600
* End of tunables for loadable kernel modules
* Added thru MR muoe951326, 04/09/95
* OS_COMPAT --- kernel compatibility options ---
* 0   Use default behavior
* 1   Use traditional behavior
* 2   Use new SVR4.2 behaviors
OS_COMPAT      1       0       2
* ------ RPC tunables muoe952511, 7/12/95 */ ---------
RPC_HIWATER    0       0       500000
RPC_LOWATER    0       0       500000
* Raw Device Async I/O parameters ---------
MAXFREE_RAIO   128     32      512
MINFREE_RAIO   32      0       128
MAXRAIO        256     32      4096
* More Raw Device Async I/O parameters ---------
RAIO_TOGGLE     1       0       1
* UnixWare-compatible async I/O parameters
NUMAIO         128     32      4096
NUMPROCAIO     32      4       1024
AIO_LISTIO_MAX 100     32      1024
AIO_MAX        1       1       1
AIO_LOCK_ON_DEMAND     0       0       1
ILDMAXOPENS    OBSOLETE
ILDMAXPPA      32      16      256
* Byn Tuneable Parameters ----------
BYN_MAXNETS    1       4       8
BYN_MAXNETWAIT 3000    9000    36000
* VNet Tuneable Parameters ----------
VNT_TRACENENT  8192    5       16384
VNT_MAXDYNCHANS        4096    2048    16384
VNT_MAXDYNGROUPS       2048    1024    8192
VNT_GSMMAX     2048    1024    8192
VNT_MRGMAX     80      32      320

**************************************

   Unix stune file

**************************************

*ident  "@(#)master:master.d/stune    1.4.1.4"
* Initial tuning performed on         Fri Jan 22 16:40:49 EST 1999
ARG_MAX 20480
BUFHWM  8192
SDATLIM 0x7fffffff
HDATLIM 0x7fffffff
SSTKLIM 0x7fffffff
HSTKLIM 0x7fffffff
SVMMLIM 0x7fffffff
HVMMLIM 0x7fffffff
DESFREE 256
MINFREE 128
LOTSFREE       512
MAXUP   400
NCALL   500
NHBUF   500
NPBUF   1024
NPROC   4000
NUMTIM  3000
NUMTRW  3000
SEMMNI  50
SEMMNS  250
SEMMNU  2500
SEMMSL  40
STRTHRESH      0x2000000
TRW_LOWATER    65535
UFSNINODE      2000
SHMMAX  2621440
MSGTQL  100
MSGMNB  65535
MSGMNI  100
MSGSEG  65535
BYN_MAXNETS    4
BYN_MAXNETWAIT 9000
KDBSYMSIZE     200000
VNT_TRACENENT  9000
VNT_MAXDYNCHANS        4096
VNT_MAXDYNGROUPS       2048
VNT_GSMMAX     2048
VNT_MRGMAX     80
SFNOLIM 1024
SANITYCLK      0
```

A.2 Database Parameters



```
Release V2R.03.00.00.04 Version 03k.j5.00.00.10
    DBSControl Utility (June 97)


The current DBS Control GDO has been read.

Enter a command, HELP, or QUIT:

DBS Control Record - General Fields:

    1.  Version                = 4
    2.  SysInit                = TRUE
    3.  DeadLockTimeout        = 240
    4.  (Reserved for future use)
    5.  HashFuncDBC            = 5 (Universal)
    6.  (Reserved for future use)
    7.  (Reserved for future use)
    8.  SessionMode            = 0 (Teradata)
    9.  LockLogger             = FALSE
    10. RollbackPriority       = FALSE
    11. MaxLoadTasks           = 5
    12. RollForwardLock        = FALSE
    13. MaxDecimal             = 18
    14. Century Break          = 0
    15. DateForm               = 0 (IntegerDate)
    16. System TimeZone Hour   = 0
    17. System TimeZone Minute = 0
    18. RollbackRSTransaction  = FALSE
    19. RSDeadLockInterval     = 0 (240)


Enter a command, HELP, or QUIT:

DBS Control Record - Performance Fields:

    1.  DictionaryCacheSize    = 128 (kilobytes)
    2.  DBSCacheCtrl           = TRUE
    3.  DBSCacheThr            = 10%
    4.  MaxParseTreeSegs       = 32
    5.  ReadAhead              = TRUE
    6.  StepsSegmentSize       = 1024 (kilobytes)
    7.  RedistBufSize          = 4 (kilobytes) (amp-level buffering
only)
    8.  DisableSyncScan        = FALSE
    9.  SyncScanCacheThr       = 10%
   10.  HTMemAlloc             = 0%
   11.  SkewAllowance          = 75%

Enter a command, HELP, or QUIT:

DBS Control Record - File System Fields:

    1.  FreeSpacePercent         = 0%
    2.  MiniCylPackLowCylProd    = 10 (free cylinders)
    3.  PermDBSize               = 63 (sectors)
    4.  JournalDBSize            = 12 (sectors)
    5.  DefragLowCylProd         = 100 (free cylinders)
    6.  PermDBAllocUnit          = 1 (sectors)
    7.  WriteDBsToDisk           = FALSE
    8.  Cylinders Saved for PERM = 10 (cylinders)


Enter a command, HELP, or QUIT:

The Internal field has been modified from FALSE to TRUE.

Enter a command, HELP, or QUIT:

DBS Control Record - Internal Fields:

    1.  BypassBynetMerge       = 0 (disabled)
    2.  DBSDiagFlags: Word 0   = 0
                      Word 1   = 0
                      Word 2   = 0
                      Word 3   = 0
    3.  DispDiagMsg            = 0
    4.  MaxAMPWorkerTasks      = 80
    5.  MaxParWorkerTasks      = 14
    6.  MaxSessions            = 120
    7.  FragSectPercent        = 25%
    8.  FSDiagFlags: Word 0    = 0
                     Word 1    = 1000
                     Word 2    = 0
                     Word 3    = 0
                     Word 4    = 0
                     Word 5    = 0
    9.  WorkDBSize             = 127 (sectors)
   10.  OptBMMaxSize           = 128 (kilobytes)
   11.  OptCPUWeight           = 1.000
   12.  OptDiskWeight          = 1.000
   13.  OptNetWeight           = 1.000
   14.  SyncScanCheckInterval  = 2048  (pages)
   15.  OptMaxNJRows           = 0
   16.  OptMinimizeAllAMPs     = FALSE
   17.  OptNoPseudoHashLocks   = FALSE
   18.  DisableSyncMerge       = FALSE
   19.  AwtStartNumber         = 80
   20.  AwtBatchSize           = 10
   21.  AwtBatchDelay          = 2
   22.  NodeBufSwitch          = 1 ( 1/2 = defaults/tune )
                               = NLB enabled in default mode.

   *** Node-level buffering is currently set to be enabled in default
mode.
```

```
   *** NLB tunables (23-31) will be ignored and their default values
used.

   23.  NodeTreeWidth          = 15 (child nodes)
   24.  DupTabBufSize          = 64 (kilobytes)
   25.  DupTabCascade          = 50%
   26.  HashedBufSize          = 32 (kilobytes)
   27.  HashedBlocking         = 100%
   28.  HashedHoldSize         = 64 (kilobytes)
   29.  HashedCascade          = 25%
   30.  HashedRecvSize         = 64 (kilobytes)
   31.  HashedRecvLimit        = 2 (buffer wraps)
   32.  EnableParserDump       = FALSE
```

Appendix B. Database Definition Statements

B.1 Create Table Statements

The following CREATE User (Database) and Table statements were used to define tables used in the TPC-D benchmark test.

```
CREATE USER tpcd100g AS PERM = 300,000,000,000 , PASSWORD = tpcd100g
AFTER JOURNAL DEFAULT JOURNAL TABLE = tpcd100g.tb_jrnl;

CREATE MULTISET TABLE LINEITEM, DATABLOCKSIZE=28.5 KILOBYTES
  (
    L_ORDERKEY       DECIMAL (15,0) not null
   ,L_PARTKEY        INTEGER not null
   ,L_SUPPKEY        INTEGER not null
   ,L_LINENUMBER     INTEGER not null
   ,L_QUANTITY       DECIMAL(15,2) not null
   ,L_EXTENDEDPRICE  DECIMAL(15,2) not null
   ,L_DISCOUNT       DECIMAL(15,2) not null
   ,L_TAX            DECIMAL(15,2) not null
   ,L_RETURNFLAG     VARCHAR(1) CASESPECIFIC not null
   ,L_LINESTATUS     VARCHAR(1) CASESPECIFIC not null
   ,L_SHIPDATE       DATE FORMAT 'yyyy-mm-dd' not null
   ,L_COMMITDATE     DATE FORMAT 'yyyy-mm-dd' not null
   ,L_RECEIPTDATE    DATE FORMAT 'yyyy-mm-dd' not null
   ,L_SHIPINSTRUCT   VARCHAR(25) CASESPECIFIC not null
   ,L_SHIPMODE       VARCHAR(10) CASESPECIFIC not null
   ,L_COMMENT        VARCHAR(44) CASESPECIFIC not null
  )
;

CREATE MULTISET TABLE ORDERTBL, DATABLOCKSIZE=29.5 KILOBYTES
  (
    O_ORDERKEY       DECIMAL (15,0) not null
   ,O_CUSTKEY        INTEGER not null
   ,O_ORDERSTATUS    VARCHAR(1) CASESPECIFIC not null
   ,O_TOTALPRICE     DECIMAL(15,2) not null
   ,O_ORDERDATE      DATE FORMAT 'yyyy-mm-dd' not null
   ,O_ORDERPRIORITY  VARCHAR(15) CASESPECIFIC not null
   ,O_CLERK          VARCHAR(15) CASESPECIFIC not null
   ,O_SHIPPRIORITY   INTEGER not null
   ,O_COMMENT        VARCHAR(79) CASESPECIFIC not null
  )
UNIQUE PRIMARY INDEX( O_ORDERKEY );

CREATE MULTISET TABLE PARTSUPP
  (
    PS_PARTKEY     INTEGER not null
   ,PS_SUPPKEY     INTEGER not null
   ,PS_AVAILQTY    INTEGER not null
   ,PS_SUPPLYCOST  DECIMAL(15,2) not null
   ,PS_COMMENT     VARCHAR(199) CASESPECIFIC not null
  )
;

CREATE MULTISET TABLE PARTTBL
  (
    P_PARTKEY      INTEGER not null
   ,P_NAME         VARCHAR(55) CASESPECIFIC not null
   ,P_MFGR         VARCHAR(25) CASESPECIFIC not null
   ,P_BRAND        VARCHAR(10) CASESPECIFIC not null
   ,P_TYPE         VARCHAR(25) CASESPECIFIC not null
   ,P_SIZE         INTEGER not null
   ,P_CONTAINER    VARCHAR(10) CASESPECIFIC not null
   ,P_RETAILPRICE  DECIMAL(15,2) not null
   ,P_COMMENT      VARCHAR(23) CASESPECIFIC not null
  )
UNIQUE PRIMARY INDEX( P_PARTKEY );

CREATE MULTISET TABLE CUSTOMER
  (
    C_CUSTKEY     INTEGER not null
   ,C_NAME        VARCHAR(25) CASESPECIFIC not null
   ,C_ADDRESS     VARCHAR(40) CASESPECIFIC not null
   ,C_NATIONKEY   INTEGER not null
   ,C_PHONE       VARCHAR(15) CASESPECIFIC not null
   ,C_ACCTBAL     DECIMAL(15,2) not null
   ,C_MKTSEGMENT  VARCHAR(10) CASESPECIFIC not null
   ,C_COMMENT     VARCHAR(117) CASESPECIFIC not null
```

```
  )
UNIQUE PRIMARY INDEX( C_CUSTKEY );

CREATE MULTISET TABLE SUPPLIER
  (
    S_SUPPKEY    INTEGER not null
   ,S_NAME       VARCHAR(25) CASESPECIFIC not null
   ,S_ADDRESS    VARCHAR(40) CASESPECIFIC not null
   ,S_NATIONKEY  INTEGER not null
   ,S_PHONE      VARCHAR(15) CASESPECIFIC not null
   ,S_ACCTBAL    DECIMAL(15,2) not null
   ,S_COMMENT    VARCHAR(101) CASESPECIFIC not null
  )
UNIQUE PRIMARY INDEX( S_SUPPKEY );

CREATE MULTISET TABLE NATION
  (
   N_NATIONKEY INTEGER NOT NULL
  ,N_NAME VARCHAR(25) CASESPECIFIC NOT NULL
  ,N_REGIONKEY INTEGER NOT NULL
  ,N_COMMENT VARCHAR(152) CASESPECIFIC NOT NULL
  )
PRIMARY INDEX ( N_NAME );

CREATE MULTISET TABLE REGION
  (
   R_REGIONKEY INTEGER NOT NULL
  ,R_NAME VARCHAR(25) CASESPECIFIC NOT NULL
  ,R_COMMENT VARCHAR(152) CASESPECIFIC NOT NULL
  )
UNIQUE PRIMARY INDEX ( R_REGIONKEY );
```

B.2 Create Index and Collect Statistics Statements

```
/*************************************************************************
\
*
*
*     NCR/Teradata TPC-D Benchmark
*
*     all Base Indexes and Statistics
*
\*************************************************************************
/

/* lineitem table: secondary indexes and column statistics
*/
collect statistics on lineitem index  (l_orderkey);
collect statistics on lineitem column l_partkey;
collect statistics on lineitem column l_suppkey;

/* customer table: secondary indexes and column statistics
*/
collect statistics customer index  (c_custkey);
create index cx_nk (c_nationkey) on customer;
collect statistics customer index cx_nk;

/*  partsupp table:  secondary indexes and column statistics         */
collect statistics partsupp index (ps_partkey);
create index psx_sk (ps_suppkey) on partsupp;
collect statistics partsupp index psx_sk;


/*     order table:  secondary indexes and column statistics         */
collect statistics ordertbl index (o_orderkey);
collect statistics on ordertbl column o_custkey;
create index ox_cl (o_clerk) on ordertbl;
collect statistics ordertbl index ox_cl;

/*     part table:  secondary indexes and column statistics          */
collect statistics parttbl index (p_partkey);
create index px_nm (p_name) on parttbl;
create index px_tp (p_type) on parttbl;
create index px_sz (p_size) on parttbl;
create index px_br_ct (p_brand,p_container) on parttbl;
collect statistics parttbl index px_nm;
collect statistics parttbl index px_tp ;
collect statistics parttbl index px_sz;
collect statistics parttbl index px_br_ct;

/*     supplier table:  secondary indexes and column statistics      */
collect statistics supplier index (s_suppkey);
create index sx_nk (s_nationkey) on supplier;
collect statistics on supplier index sx_nk;

/*     region table:  secondary indexes and column statistics        */
collect statistics region index (r_regionkey);
create index rx_nm (r_name) on region;
collect statistics on region index rx_nm;

/*     nation table:  column statistics                              */
collect statistics nation index (n_name);
collect statistics on nation column n_regionkey;
create index nx_nk (n_nationkey) on nation;
collect statistics on nation index nx_nk;

alter table lineitem, default datablocksize;
alter table ordertbl, default datablocksize;

/*************************************************************************
\
*
*
*     NCR/Teradata TPC-D Benchmark
*
*     Join Index j1 and Statistics
*
*
*
\*************************************************************************
/
```

```
create join index j1 as select
(l_orderkey, o_orderdate, c_nationkey, o_shippriority, o_custkey,
o_orderpriority, c_mktsegment),
(l_partkey, l_suppkey, s_nationkey, l_shipdate, l_quantity,
l_extendedprice,
l_discount, l_returnflag, l_commitdate, l_receiptdate, l_shipmode)
from (lineitem left join supplier on l_suppkey=s_suppkey)
left join
(ordertbl left join customer on c_custkey=o_custkey) on
l_orderkey=o_orderkey order by o_orderdate;

create index (l_orderkey) on j1;

collect statistics on j1 index (l_orderkey);
collect statistics on j1 column o_orderdate;
collect statistics on j1 column c_nationkey;
collect statistics on j1 column l_partkey;
collect statistics on j1 column l_suppkey;
collect statistics on j1 column s_nationkey;

create index c1 all
(l_partkey, l_suppkey, l_shipdate, l_quantity, l_extendedprice,
l_discount)
order by (l_shipdate) on j1;

/*************************************************************************
\
*
*
*     NCR/Teradata TPC-D Benchmark
*
*     Join Index j2 and Statistics
*
*
*
\*************************************************************************
/

create join index j2 as select
(l_partkey, o_orderdate, l_orderkey, c_nationkey, o_custkey),
(l_suppkey, s_nationkey, l_quantity, l_extendedprice, l_discount)
from (lineitem left join supplier on l_suppkey=s_suppkey)
left join
(ordertbl left join customer on c_custkey=o_custkey) on
l_orderkey=o_orderkey;

collect statistics on j2 column l_partkey;
collect statistics on j2 column o_orderdate;

/*************************************************************************
\
*
*
*     NCR/Teradata TPC-D Benchmark
*
*     Join Index j3 and Statistics
*
*
*
\*************************************************************************
/

create join index j3 as select
(ps_suppkey,s_nationkey),
(ps_partkey,ps_supplycost,ps_availqty)
from partsupp left join supplier on ps_suppkey=s_suppkey;

collect statistics on j3 index (ps_suppkey);
create index (s_nationkey) on j3;
collect statistics on j3 index (s_nationkey);

/*************************************************************************
\
*
*
*     NCR/Teradata TPC-D Benchmark
*
*     Join Index j4 and Statistics
*
*
*
\*************************************************************************
/

create join index j4 as select
ps_suppkey, p_size, parttbl.rowid, p_type, p_mfgr, p_brand, ps_partkey,
ps_supplycost
from (partsupp left outer join parttbl on p_partkey=ps_partkey)
order by p_size asc
primary index (ps_suppkey);

collect statistics j4 index (ps_suppkey);
collect statistics j4 column p_size;

/*************************************************************************
\
*
*
*     NCR/Teradata TPC-D Benchmark
*
*     Join Index j5 and Statistics
*
*
*
\*************************************************************************
/

create join index j5 as select
l_shipdate, l_returnflag, l_linestatus,
sum(l_discount (float)) as sum_disc,
sum(l_quantity (float)) as sum_qty,
sum(l_extendedprice (float)) as sum_base_price,
sum(l_extendedprice*(1-l_discount) (float))  as sum_disc_price,
sum(l_extendedprice*(1-l_discount)*(1+l_tax) (float)) as sum_charge,
count(*) (float) as count_order
```

```
from lineitem
group by l_returnflag, l_linestatus, l_shipdate
order by l_shipdate;

/*************************************************************************
\
*
*
*     NCR/Teradata TPC-D Benchmark
*
*     Join Index j6 and Statistics
*
*
*
\*************************************************************************
/

create join index j6 as select
extract(year from l_shipdate) as l_year, l_discount, l_quantity,
sum(l_extendedprice*l_discount (float)) as revenue
from lineitem
group by l_year,l_discount,l_quantity
order by l_year primary index(l_discount,l_quantity);

/*************************************************************************
\
*
*
*     NCR/Teradata TPC-D Benchmark
*
*     Join Index j7 and Statistics                          *
*
*
\*************************************************************************
/

create join index j7 as select
extract(year from o_orderdate) as o_year, s_nationkey,
sum(l_extendedprice*(1-l_discount) (float)) as revenue
from customer, ordertbl, lineitem, supplier
where
c_custkey = o_custkey
and o_orderkey = l_orderkey
and l_suppkey = s_suppkey
and c_nationkey = s_nationkey
group by o_year, s_nationkey
order by o_year primary index (s_nationkey);

/*************************************************************************
\
*
*
*     NCR/Teradata TPC-D Benchmark
*
*     Join Index j8 and Statistics
*
*
*
\*************************************************************************
/

create join index j8 as select
extract(year from l_shipdate) as l_year, s_nationkey, c_nationkey,
sum(l_extendedprice * (1-l_discount) (float)) as revenue
from supplier, lineitem, ordertbl, customer
where
s_suppkey = l_suppkey
and o_orderkey = l_orderkey
and c_custkey = o_custkey
group by l_year, s_nationkey, c_nationkey
order by l_year primary index (s_nationkey, c_nationkey);

/*************************************************************************
\
*
*
*     NCR/Teradata TPC-D Benchmark
*
*     Join Index j9 and Statistics
*
*
*
\*************************************************************************
/

create join index j9 as select
extract(year from o_orderdate) as o_year, c_nationkey, s_nationkey,
p_type,
sum(l_extendedprice*(1-l_discount) (float)) as mkt_share
from parttbl, supplier, lineitem, ordertbl, customer
where
p_partkey = l_partkey
and s_suppkey = l_suppkey
and l_orderkey = o_orderkey
and o_custkey = c_custkey
group by o_year,c_nationkey,s_nationkey,p_type
order by o_year primary index (c_nationkey,s_nationkey,p_type);

/*************************************************************************
\
*
*
*     NCR/Teradata TPC-D Benchmark
*
*     Join Index j10 and Statistics
*
*
*
\*************************************************************************
/

create join index j10 as select
  l_shipmode,o_orderpriority,
  extract (year from l_receiptdate) as l_year
  from ordertbl, lineitem
  where o_orderkey = l_orderkey
```

```
  and l_commitdate < l_receiptdate
  and l_shipdate < l_commitdate
  group by l_shipmode, o_orderpriority, l_year
  order by l_year
  primary index (l_shipmode, o_orderpriority, l_year);

/*************************************************************************
\
*
*
*     NCR/Teradata TPC-D Benchmark
*
*     Join Index j11 and Statistics
*
*
*
\*************************************************************************
/

create join index j11 as select
extract (year from l_shipdate) as l_year,
extract (month from l_shipdate) as l_month,
p_type, sum(l_extendedprice*(1-l_discount) (float)) as promo_rev
from lineitem,parttbl
where l_partkey = p_partkey
group by l_year,l_month,p_type
primary index (l_year,l_month,p_type);
```

Appendix C. Query Text and Output Data from Qualification Database

```
/*    'TPC-D Pricing Summary Report Query (Q1)'                    */


SELECT
  L_RETURNFLAG (FORMAT 'x(1)'), L_LINESTATUS (FORMAT 'x(1)'),
  SUM(L_QUANTITY (FLOAT)) (DECIMAL(18,2)) AS SUM_QTY,
  SUM(L_EXTENDEDPRICE (FLOAT)) (DECIMAL(18,2)) AS SUM_BASE_PRICE,
  SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT) (FLOAT)) (DECIMAL(18,2)) AS
SUM_DISC_PRICE,
  SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)*(1+L_TAX) (FLOAT))
  (DECIMAL(18,2)) AS SUM_CHARGE,
  AVG(L_QUANTITY (FLOAT)) (DECIMAL(18,2)) AS AVG_QTY,
  AVG(L_EXTENDEDPRICE (FLOAT)) (DECIMAL(18,2)) AS AVG_PRICE,
  AVG(L_DISCOUNT (FLOAT)) (DECIMAL(18,2)) AS AVG_DISC,
  COUNT(*) AS COUNT_ORDER
  FROM LINEITEM
  WHERE
  L_SHIPDATE <= ('1998-12-01' (DATE, FORMAT 'yyyy-mm-dd')) - 90 (DATE)
  GROUP BY L_RETURNFLAG, L_LINESTATUS
  ORDER BY L_RETURNFLAG, L_LINESTATUS;


 Query 1 complete, 4 rows returned


|Col1|Col2|_____________________Col3|_____________________Col4|_____________
___Col5|_____________________Col6|_____________________Col7|_________________
__Col8|_____________________Col9|______Col10|
|A   |F   |             3773034.00|          5319329289.68|
5053976845.78|          5256336547.68|                  25.51|
35964.01|                  0.05|     147907|
|N   |F   |              100245.00|           141459686.10|
134380852.77|          139710306.87|                 25.62|
36160.45|                  0.05|       3912|
|N   |O   |             7464940.00|         10518546073.98|
9992072944.46|         10392414192.06|                  25.54|
35990.13|                  0.05|     292262|
|R   |F   |             3779140.00|          5328886172.99|
5062370635.93|          5265431221.82|                  25.55|
36025.46|                  0.05|     147920|


 Submitting SQL Request #2:

/*    'TPC-D Minimum Cost Supplier (Q2)'                           */


SELECT
  S_ACCTBAL, S_NAME (FORMAT 'x(25)'), N_NAME (FORMAT 'x(25)'),
P_PARTKEY,
  P_MFGR (FORMAT 'x(25)'), S_ADDRESS, S_PHONE (FORMAT 'x(15)'),
S_COMMENT
  FROM PARTTBL, SUPPLIER, PARTSUPP, NATION, REGION
  WHERE
  P_PARTKEY = PS_PARTKEY
  AND S_SUPPKEY = PS_SUPPKEY
  AND P_SIZE = 15
  AND P_TYPE LIKE '%BRASS'
  AND S_NATIONKEY = N_NATIONKEY
  AND N_REGIONKEY = R_REGIONKEY
  AND R_NAME = 'EUROPE'
  AND PS_SUPPLYCOST =
  (SELECT MIN(PS_SUPPLYCOST)
  FROM PARTSUPP, SUPPLIER, NATION, REGION
  WHERE
```

```
  P_PARTKEY = PS_PARTKEY
  AND S_SUPPKEY = PS_SUPPKEY
  AND S_NATIONKEY = N_NATIONKEY
  AND N_REGIONKEY = R_REGIONKEY
  AND R_NAME = 'EUROPE')
  ORDER BY S_ACCTBAL DESC, N_NAME, S_NAME, P_PARTKEY;


 Query 2 complete, 44 rows returned


|_____________Col1|Col2_____________________|Col3_____________________
_|_______Col4|Col5_____________________|Col6_____________________________
_________|Col7__________|Col8_____________________________________
_____________________________|
|            9828.21|Supplier#000000647       |UNITED KINGDOM
|      13120|Manufacturer#5           |jB16PyPyB7Bl52jMjSPw3mS
|33-258-202-4782|z1QhSiMj11Bm7COlLwh6Ql0B1R2Mg4CLn
LhiP0wiMzy72hlkP7l5in2y6RS6N130lz51nSRL5gOg5S26hPCCQN2L|
|            9508.37|Supplier#000000070       |FRANCE
|       3563|Manufacturer#1           |M5C616R5h5SlMR3zzmLkSw24j2
|16-821-608-1166|m7z0CPShmBkhlChBAi3LkQ2CLw
mhl6QP362RPS3044CB2y41yhOhjlBin0CL7yhxmhS4hBM07kQ1yyjOjz3C|
|            9508.37|Supplier#000000070       |FRANCE
|      17268|Manufacturer#4           |M5C616R5h5SlMR3zzmLkSw24j2
|16-821-608-1166|m7z0CPShmBkhlChBAi3LkQ2CLw
mhl6QP362RPS3044CB2y41yhOhjlBin0CL7yhxmhS4hBM07kQ1yyjOjz3C|
|            9453.01|Supplier#000000802       |ROMANIA
|      10021|Manufacturer#5
|5yARQNSLNRAlOlBnkNQCik3SOlyClk7nmRhA2h0 |29-342-882-
6463|65y3RQ2i0OP6Nz7mS hC
PxwLy7L1jQy6Ol63xO3iBCz52Rm1zm0MziCMLij2n6wky51mBOwx Qh52iz
QB1545Amxyj|
|            9453.01|Supplier#000000802       |ROMANIA
|      13275|Manufacturer#4
|5yARQNSLNRAlOlBnkNQCik3SOlyClk7nmRhA2h0 |29-342-882-
6463|65y3RQ2i0OP6Nz7mS hC
PxwLy7L1jQy6Ol63xO3iBCz52Rm1zm0MziCMLij2n6wky51mBOwx Qh52iz
QB1545Amxyj|
|            9192.10|Supplier#000000115       |UNITED KINGDOM
|      13325|Manufacturer#1
|h0m3lzlSPMw2B0ny7LNyNCkjRRn7iyMlLBLA    |33-597-248-1220|1QzQjhSyx
ixm2lgz2Ry7075RL3MS5z36x56hxmR0wLN0LBxm164LzCMmALzOAjn4kz7i4wjOlCONl1C5
1M7nCMx66SBRAQA|
|            9032.15|Supplier#000000959       |GERMANY
|       4958|Manufacturer#4           |205LNCzxMCnQ5gnz4n S3ynP6Mhnw
|17-108-642-3106|Px z7kOx56l7jQz NwBBQhky yM7kLgxRQw5zw6 426Bm55lC6
OkQ7hQPLixjM7y47BNP16CRi0kjk354lgxh|
|            8702.02|Supplier#000000333       |RUSSIA
|      11810|Manufacturer#3
|5iwkgN5n2BN15OmQk2602h0N6NzxPyiPN5lnj    |32-508-202-
6136|SgimAjmn3wL7Rlxmh3LCwOPnhjyl 7xxzxAN 4ACx43y65NwQ7P
|
|            8615.50|Supplier#000000812       |FRANCE
|      10551|Manufacturer#2           |h4i2M2O0
ky1g2mlBOmxjzj0hA2h6nkSNhP     |16-585-724-
6633|57i0NAyR0RP2jOh54C6B22OlSL
|
|            8615.50|Supplier#000000812       |FRANCE
|      13811|Manufacturer#4           |h4i2M2O0
ky1g2mlBOmxjzj0hA2h6nkSNhP     |16-585-724-
6633|57i0NAyR0RP2jOh54C6B22OlSL
|
|            8488.53|Supplier#000000367       |RUSSIA
|       6854|Manufacturer#4           |nkmQ2Qzgh0wA 3x Sn2S7N5gmSOj xwC
COSn6  |32-458-198-9557|35C2RROP C  Nlgi2N
SxAj0hQkn7kP5z4wSxSwgMxj6k4MRmh0S2Qm7R3z4jB OOQBMl          |
|            8430.52|Supplier#000000646       |FRANCE
|      11384|Manufacturer#3           |61SjP6S y B0 32l11
|16-601-220-5489|kiw4NSNBNxy5kywzwyx0PMM21xiMOhxR423Akkm
Q7CNwRzQS23Nzz22 mnm6P377Q3Mj7n 56BLm6lxwllh kSmN|
|            8271.39|Supplier#000000146       |RUSSIA
|       4637|Manufacturer#5           |wh yPSk6hNBlB4l33iQ0wS0
RhBhQ4zQ3lz     |32-792-619-3155|jjwgwljRO63
n7OM2MP0hg3L1mlwBMLmMlS4Cgyn LA5PwC2P0AS6g3C5mkOjO72NPig731m        |
|            8096.98|Supplier#000000574       |RUSSIA
|        323|Manufacturer#4           |hCOj4Cgx43xx jgP4QkL7gLN65
|32-866-246-8752|OhxNj6SlB56315B3k5SCBzwQyLk76zlj4Ow2Q
BC2wACkxh3S0RCyx6nARzSQR2010k0BCPhOg6yQm  |
|            7392.78|Supplier#000000170       |UNITED KINGDOM
|       7655|Manufacturer#2           |PCxjjzNQihLNxgLw0SiMmQ
|33-803-340-5398|M116S1xzg54iC3k7OPLQi3Cimhghz2BClQk
g5Ag12QSBlhglANnw4MR MBS 72A              |
|            7205.20|Supplier#000000477       |GERMANY
|      10956|Manufacturer#5           |Mimj6403h zmAzAgg Bjy05O 2z
|17-180-144-7991|yRlyR SnMxmhPjAmBw
S02AxQ6yOhBRlOwzmlxzO0A2Sx075kjlAknn7z2 O0S7hy0BiknwOQm6Pmz3gL4gj2z7|
|            6820.35|Supplier#000000007       |UNITED KINGDOM
|      13217|Manufacturer#5           |z45m2jBRzl5ilLNz4
|33-990-965-2201|1PhngjmiSQl0RzRACP0l4S70xSL
QPSBM16072SkMLCgm4OOMjARLNQk3g1P3BB32AgBMl462B0CP7Rh24|
|            6721.70|Supplier#000000954       |FRANCE
|       4191|Manufacturer#3           |OM7xnNxNnkgQ mzh2g3RQmg1g
|16-537-341-8517|5ni3yCkmz5ymx0kCg74zhLA B516Si1w152AkiByx1Nl
NgghAkkmNz1jASj4mxzxznOySg7hAyM3MRRnBj|
|            6329.90|Supplier#000000996       |GERMANY
|      10735|Manufacturer#2
|k6135gA3zPwNl7L3Rl45mlnACjngOQQBB300iyA |17-447-811-
3282|PBO7wjlQMm1h3AAA 1NQAl0kkijnkRNgQ0 mh1z6QS0gC5lP1
ykmzNR20OlN506ARSOz3j          |
|            6173.87|Supplier#000000408       |RUSSIA
|      18139|Manufacturer#1           |Cni6 zR5C4lh104POx5h05
mg53CQ2Sw4SAM2M2x|32-858-724-
2950|10SxMOwhjON3khzQ124gNnyw7B4nL7ml4L5llSR
|
|            5364.99|Supplier#000000785       |RUSSIA
|      13784|Manufacturer#4           |71OnPzQkC2P1hRNRgjyQP4n1
|32-297-653-2203|kiiPQ3ik7R ykAhRx43Rw70L1Ok
7AMi3AjRw7lklwxwyiL6S2O1COyS4QB46m5M167mjMwCm0w     |
|            5069.27|Supplier#000000328       |GERMANY
|      16327|Manufacturer#1           |5O4033xSgml
|17-231-513-5721| OMk3ALAPNmj6lBLMAS7M1nCAS
4xLj51iy2klix3nPi26gAxPgANmk6zSi6 3A7m l11BOwiC6xLB4hBRiPM|
|            4941.88|Supplier#000000321       |ROMANIA
|       7320|Manufacturer#5           |hyLQ mg42S2kAMlj M3BwMSjS
```

```
|29-573-279-1406|y2644kMhOkPCm5P5y7Lmz7OR6mgSmBN631RggmC
|
|             4672.25|Supplier#000000239       |RUSSIA
|      12238|Manufacturer#1           |y4ymj7B5BN1nMSkwPPggAl
|32-396-654-6826|Py3RA2gykmSCmj0z3ii7Rxzhz6OyR RxS C3S23LPQ
|
|             4586.49|Supplier#000000680       |RUSSIA
|       5679|Manufacturer#3           |BP1Nlw5nPMxRnOAwm
|32-522-382-1620|kA0y25RNO1Al im7SyiPzSym3M5OS52l6S576kn0S2k
0mPBLlAzL6Ax7CM6iNi4CgCy6Bln7hlhxmlRng |
|             4518.31|Supplier#000000149       |FRANCE
|      18344|Manufacturer#5           |4B QSy5Bl2
|16-660-553-2456|hijkPhgL1g4L1Q27y0Q42wh0Qz3jPiL4NgkM4NNg1
llQ1yNNBk1C1QnlRO7 4ki                |
|             4315.15|Supplier#000000509       |FRANCE
|      18972|Manufacturer#2           |B5 iPRn7L4yMllgwCnRPMA
|16-298-154-3365|ygiPh7ymP7jBznmR2lQLLgjmi1wik
|
|             3526.53|Supplier#000000553       |FRANCE
|       8036|Manufacturer#4           |ylLOx2gMw 5iBl6AiNL60Q
|16-599-552-3755|L3ggShlRlyxmR4MNl7Rw7OQign6yO
|
|             3526.53|Supplier#000000553       |FRANCE
|      17018|Manufacturer#3           |ylLOx2gMw 5iBl6AiNL60Q
|16-599-552-3755|L3ggShlRlyxmR4MNl7Rw7OQign6yO
|
|             3294.68|Supplier#000000350       |GERMANY
|       4841|Manufacturer#4           |x5kRL2z1BPg0 BO 2hi1iOyh
30RRg0OPj      |17-113-181-4017|BjQznni44OmQ7S16yl3zxk2M6nM4M 27yMPML
|
|             2972.26|Supplier#000000016       |RUSSIA
|       1015|Manufacturer#4           |B7wLkSLRjNS MS1C
|32-822-502-4215|C7w6S6QzhAPQmMmNmMN1hA0llQOA 00m1NmC25wyQ461SA
jy03zmRh22MLM00zhmi             |
|             2963.09|Supplier#000000840       |ROMANIA
|       3080|Manufacturer#2           |1ynwiQkNh0
CMRRCk4l306M2ij0jykg6QNgSCAzy|29-781-337-
5584|S7NRMx43RmOjxML6hxLyN75LzxBwB0wjSLx3 S3Cwh52S6ilSOLhQm0 6C1
yzx3jPm6Sjg 5By0BCPwOR32i1CQgxR0gB43gh|
|             2221.25|Supplier#000000771       |ROMANIA
|      13981|Manufacturer#2           |LAjCRjl3nAMzzhmw0Sx1Mg
|29-986-304-9006|jhk0N7NlhS23iCngC52BBC
0jilM0wByx0LB5R070R2lCx113lQiS7xNhBRA0xknlNxLiA          |
|             1381.97|Supplier#000000104       |FRANCE
|      18103|Manufacturer#3           |i Qnl4 1 jiwM
C2yxAylL5R4SBQh54N6       |16-434-972-
6922|MwnBw1g71Pig2Am7nz0Mm5SNl7OwQLAkN56ji
|
|              906.07|Supplier#000000138       |ROMANIA
|       8363|Manufacturer#4           |1iBxSxLl1Mh3
6LS6PlLPNlnlMjCQh22z6n5    |29-533-434-6776|nLjQAmCw77R2jRMgz5LSyyx1QN
l 4jMMO3RAkxOkzRmwQl3Qm5236k72RRPnim0 BkzQnBMM6A  PMml2n|
|              765.69|Supplier#000000799       |RUSSIA
|      11276|Manufacturer#2           |Am7yihz47mg NkgQL w By4
|32-579-339-1495|MMRPNQ 4l66mQQPNniAiiL0PQ2C4yyBRn1nRlxnxkj5Ak45Pw
mQk1ROhz66BRQiiL gPRQRy 56MyQ nS1N14R 7Ml6xhl2lOS3|
|             727.89|Supplier#000000470       |ROMANIA
|       6213|Manufacturer#3           |gAySBM2N7 PgwP5kiP4n7BzOik0M
|29-165-289-1523|zCkPgn
6wN5A3R47gljlQ3hNSLShP2RALxCiinkOy4wCwA1LCiBO5yiSC yBAA lii
|
|              683.07|Supplier#000000651       |RUSSIA
|       4888|Manufacturer#4
|ymQ6PByCh4lzxBBPLB2wwOhRh47wQMQSPL      |32-181-426-
4490|kx6jhQkwzg6RkRgPLPM30BgL1R726l1m5AMk0MmMQBQ
nCihiLxhMgCgRih6MmMx0PglRQ7AQnl72g50|
|             167.56|Supplier#000000290       |FRANCE
|       2037|Manufacturer#1           |xm6n30QmBn75QPh7Py01lxlnB4n1lNh
MBi1   |16-675-286-5102|gi6y41Bg5
AkhmCh30Qyxx515xLi6MRjCQ4n2xjML7N0PNSCyBPwS1C                          |
|              91.39|Supplier#000000949       |UNITED KINGDOM
|       9430|Manufacturer#2           |zCmkwm43LQ62Q3O 4nkNA
|33-332-697-2768|Nm5yn23SxgPAMRz6B5CjAj04nkNx5lA7O
|
|             -314.06|Supplier#000000510       |ROMANIA
|      17242|Manufacturer#4           |hwx1kP6nQ0NgPO232nxAwL4QPjRN
i64jOiPjC0 |29-207-852-
3454|5NAyRmwSCQ5hMB6RiN0Qyhjghnl0BPjPlNlP725AAOCw
|
|             -820.89|Supplier#000000409       |GERMANY
|       2156|Manufacturer#5           |Nk4z1lg25gNwMwO
2BnMOn1P3k0yA7i2l       |17-719-517-9836|CmLnngQywQ4hhx3266yz5Qgj
nOm1P1RC3gxmg4kmnNCSnM4PBO4hNQA1xyzC2LCQBP07w2z3      |
|             -845.44|Supplier#000000704       |ROMANIA
|       9926|Manufacturer#5           |zi jjmO4A57 mQgLmhALCM3B zL
|29-300-896-5991|OSh0PRm6ByyS2wLBNPQPPi2CL0zN0Px1LNOlwl6y0Rgij zP5hPCz
745nn62hPORl6i7SynBCC4Qy4Qkn1 l|
|             -942.73|Supplier#000000563       |GERMANY
|       5797|Manufacturer#1           |xSLLy 4jm17OC
|17-108-537-2691|1gSOl3A0jnkAmQLllmygzmg7hhjjAmRxN5SwjxM16Ok61 wm5N42
|
```

Submitting SQL Request #3:

```
/*    'TPC-D Shipping Priority Query (Q3)'                          */



SELECT
  L_ORDERKEY,
  SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT) (FLOAT)) (DECIMAL(18,2)) AS
REVENUE,
  O_ORDERDATE, O_SHIPPRIORITY
  FROM  CUSTOMER, ORDERTBL, LINEITEM
  WHERE
  C_MKTSEGMENT  = 'BUILDING'
  AND C_CUSTKEY    = O_CUSTKEY
  AND L_ORDERKEY   = O_ORDERKEY
  AND O_ORDERDATE  < '1995-03-15'
  AND L_SHIPDATE   > '1995-03-15'
  GROUP BY L_ORDERKEY, O_ORDERDATE, O_SHIPPRIORITY
  ORDER BY REVENUE DESC, O_ORDERDATE;
```

```
 Query 3 complete, 1198 rows returned
 only displaying 10 rows per TPC-D spec
```

```
|_____________Col1|_____________________Col2|_____Col3|_______Col4|
|          260930.00|              320547.25|1995-03-12|          0|
|          402497.00|              298879.53|1995-02-12|          0|
|          457859.00|              296490.68|1995-01-17|          0|
|          509889.00|              294068.87|1995-02-03|          0|
|           58117.00|              292632.83|1995-02-21|          0|
|          538311.00|              279666.00|1995-03-07|          0|
|          588421.00|              275477.12|1995-03-03|          0|
|          416167.00|              273765.45|1995-02-22|          0|
|           97830.00|              273227.06|1995-03-04|          0|
|           90276.00|              272233.92|1995-03-04|          0|
```

```
 Submitting SQL Request #4:
```

```
/* 'TPC-D Order Priority Checking Query (Q4)'                    */
```

```
SELECT
  O_ORDERPRIORITY (FORMAT 'x(15)'), COUNT(*) AS ORDER_COUNT
  FROM ORDERTBL
  WHERE
  O_ORDERDATE >=  '1993-07-01'
  AND O_ORDERDATE < ADD_MONTHS('1993-07-01',3)
  AND EXISTS (SELECT *
  FROM LINEITEM
  WHERE
  L_ORDERKEY = O_ORDERKEY
  AND L_COMMITDATE < L_RECEIPTDATE)
  GROUP BY O_ORDERPRIORITY
  ORDER BY O_ORDERPRIORITY;
```

```
 Query 4 complete, 5 rows returned
```

```
|Col1_____________|_______Col2|
|1-URGENT         |        999|
|2-HIGH           |       1002|
|3-MEDIUM         |       1021|
|4-NOT SPECIFIED  |        997|
|5-LOW            |       1089|
```

```
 Submitting SQL Request #5:
```

```
/* 'TPC-D Local Supplier Volume Query (Q5)'                      */
```

```
SELECT
  N_NAME (FORMAT 'x(25)'),
  SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT) (FLOAT)) (DECIMAL(18,2)) AS
REVENUE
  FROM CUSTOMER, ORDERTBL, LINEITEM, SUPPLIER, NATION, REGION
  WHERE
  C_CUSTKEY = O_CUSTKEY
  AND O_ORDERKEY = L_ORDERKEY
  AND L_SUPPKEY = S_SUPPKEY
  AND C_NATIONKEY = S_NATIONKEY
  AND S_NATIONKEY = N_NATIONKEY
  AND N_REGIONKEY = R_REGIONKEY
  AND R_NAME = 'ASIA'
  AND O_ORDERDATE >= '1994-01-01'
  AND O_ORDERDATE < ADD_MONTHS('1994-01-01',12)
  GROUP BY N_NAME
  ORDER BY REVENUE DESC;
```

```
 Query 5 complete, 5 rows returned
```

```
|Col1_____________________|_________________Col2|
|CHINA                    |          7349391.47|
|INDONESIA                |          6485853.40|
|INDIA                    |          5505346.82|
|JAPAN                    |          5388883.59|
|VIETNAM                  |          4728846.60|
```

```
 Submitting SQL Request #6:
```

```
/* 'TPC-D Forecasting Revenue Change Query (Q6)'                 */
```

```
SELECT
  SUM(L_EXTENDEDPRICE*L_DISCOUNT (FLOAT)) (DECIMAL(18,2)) AS REVENUE
  FROM LINEITEM
  WHERE
  L_SHIPDATE >= '1994-01-01'
  AND L_SHIPDATE <  ADD_MONTHS('1994-01-01',12)
  AND L_DISCOUNT BETWEEN .06 - 0.01 AND .06 + 0.01
  AND L_QUANTITY < 24;
```

```
 Query 6 complete, 1 rows returned
```

```
|___________________Col1|
```

```
|           11450588.04|
```

```
 Submitting SQL Request #7:

/* 'TPC-D Volume Shipping Query (Q7)'                                */
```

```
SELECT
  N1.N_NAME (FORMAT 'x(25)') AS SUPP_NATION,
  N2.N_NAME (FORMAT 'x(25)') AS CUST_NATION,
  EXTRACT(YEAR FROM L_SHIPDATE) AS "YEAR",
  SUM(L_EXTENDEDPRICE * (1-L_DISCOUNT) (FLOAT)) (DECIMAL(18,2)) AS
REVENUE
  FROM SUPPLIER, LINEITEM, ORDERTBL, CUSTOMER, NATION N1, NATION N2
  WHERE
  S_SUPPKEY  = L_SUPPKEY
  AND O_ORDERKEY = L_ORDERKEY
  AND C_CUSTKEY = O_CUSTKEY
  AND S_NATIONKEY = N1.N_NATIONKEY
  AND C_NATIONKEY = N2.N_NATIONKEY
  AND ((N1.N_NAME = 'FRANCE' AND N2.N_NAME = 'GERMANY')
  OR (N1.N_NAME = 'GERMANY' AND N2.N_NAME = 'FRANCE'))
  AND  L_SHIPDATE BETWEEN '1995-01-01' AND '1996-12-31'
  GROUP BY SUPP_NATION, CUST_NATION, "YEAR"
  ORDER BY SUPP_NATION, CUST_NATION, "YEAR";
```

```
 Query 7 complete, 4 rows returned

|Col1_____________________|Col2_____________________|_______Col3|______
_____________Col4|
|FRANCE                   |GERMANY                  |       1995|
4611421.44|
|FRANCE                   |GERMANY                  |       1996|
4828420.37|
|GERMANY                  |FRANCE                   |       1995|
6755766.84|
|GERMANY                  |FRANCE                   |       1996|
5810951.40|
```

```
 Submitting SQL Request #8:

/* 'TPC-D National Market Share Query (Q8)'                          */
```

```
SELECT
  EXTRACT(YEAR FROM O_ORDERDATE) AS "YEAR",
  SUM(CASE WHEN N2.N_NAME = 'BRAZIL' THEN (L_EXTENDEDPRICE*(1-
L_DISCOUNT) (FLOAT))
  ELSE 0 END) / SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT) (FLOAT))
  (DECIMAL(18,2)) AS MKT_SHARE
  FROM PARTTBL, SUPPLIER, LINEITEM, ORDERTBL, CUSTOMER,
  NATION N1, NATION N2, REGION
  WHERE
  P_PARTKEY = L_PARTKEY
  AND S_SUPPKEY = L_SUPPKEY
  AND L_ORDERKEY = O_ORDERKEY
  AND O_CUSTKEY = C_CUSTKEY
  AND C_NATIONKEY = N1.N_NATIONKEY
  AND N1.N_REGIONKEY = R_REGIONKEY
  AND R_NAME = 'AMERICA'
  AND S_NATIONKEY = N2.N_NATIONKEY
  AND O_ORDERDATE BETWEEN '1995-01-01' AND '1996-12-31'
  AND P_TYPE = 'ECONOMY ANODIZED STEEL'
  GROUP BY "YEAR"
  ORDER BY "YEAR";
```

```
 Query 8 complete, 2 rows returned

|_______Col1|____________________Col2|
|       1995|                    0.05|
|       1996|                    0.09|
```

```
 Submitting SQL Request #9:

/* 'TPC-D Product Type Profit Measure Query (Q9)'                    */
```

```
SELECT
  N_NAME (FORMAT 'x(25)') AS NATION, EXTRACT(YEAR FROM O_ORDERDATE) AS
"YEAR",
  SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)-PS_SUPPLYCOST*L_QUANTITY (FLOAT))
  (DECIMAL(18,2)) AS SUM_PROFIT
  FROM PARTTBL, SUPPLIER, LINEITEM, PARTSUPP, ORDERTBL, NATION
  WHERE
  S_SUPPKEY = L_SUPPKEY
  AND PS_SUPPKEY = L_SUPPKEY
  AND PS_PARTKEY = L_PARTKEY
  AND P_PARTKEY = L_PARTKEY
  AND O_ORDERKEY = L_ORDERKEY
  AND S_NATIONKEY = N_NATIONKEY
  AND P_NAME LIKE '%green%'
  GROUP BY NATION, "YEAR"
  ORDER BY NATION, "YEAR" DESC;
```

```
 Query 9 complete, 175 rows returned
```

Col1	Col2	Col3
ALGERIA	1998	1946316.01
ALGERIA	1997	2973825.69
ALGERIA	1996	3308881.52
ALGERIA	1995	3092227.30
ALGERIA	1994	3406958.71
ALGERIA	1993	3140744.03
ALGERIA	1992	3330704.41
ARGENTINA	1998	3045410.01
ARGENTINA	1997	4255378.59
ARGENTINA	1996	4651751.94
ARGENTINA	1995	4897797.00
ARGENTINA	1994	4823465.77
ARGENTINA	1993	4499810.71
ARGENTINA	1992	4764593.39
BRAZIL	1998	2932051.36
BRAZIL	1997	3784531.35
BRAZIL	1996	3965665.69
BRAZIL	1995	4063060.86
BRAZIL	1994	4236277.35
BRAZIL	1993	4363461.31
BRAZIL	1992	4684749.23
CANADA	1998	2217064.04
CANADA	1997	2950110.61
CANADA	1996	3184049.97
CANADA	1995	3962540.19
CANADA	1994	3365251.02
CANADA	1993	3617013.37
CANADA	1992	3407955.25
CHINA	1998	3048192.02
CHINA	1997	5001207.69
CHINA	1996	4800958.31
CHINA	1995	5154927.73
CHINA	1994	5882634.53
CHINA	1993	4733364.82
CHINA	1992	5014704.08
EGYPT	1998	1892538.74
EGYPT	1997	3849220.07
EGYPT	1996	3418656.55
EGYPT	1995	3766170.60
EGYPT	1994	3520025.56
EGYPT	1993	4375424.75
EGYPT	1992	4586034.39
ETHIOPIA	1998	1860117.73
ETHIOPIA	1997	3705722.33
ETHIOPIA	1996	3577215.39
ETHIOPIA	1995	3425219.55
ETHIOPIA	1994	3428616.18
ETHIOPIA	1993	3459815.43
ETHIOPIA	1992	3280072.91
FRANCE	1998	1592531.55
FRANCE	1997	2746176.54
FRANCE	1996	2505844.88
FRANCE	1995	2902077.00
FRANCE	1994	2532229.56
FRANCE	1993	2305725.44
FRANCE	1992	2955126.69
GERMANY	1998	3538625.73
GERMANY	1997	4425943.40
GERMANY	1996	4266344.96
GERMANY	1995	3952963.52
GERMANY	1994	4462655.80
GERMANY	1993	4435094.66
GERMANY	1992	4521715.41
INDIA	1998	3378369.34
INDIA	1997	4186477.85
INDIA	1996	5074383.93
INDIA	1995	4487435.38
INDIA	1994	4718312.63
INDIA	1993	4499573.81
INDIA	1992	4712930.33
INDONESIA	1998	2902077.10
INDONESIA	1997	4973644.23
INDONESIA	1996	4977652.49
INDONESIA	1995	5359380.15
INDONESIA	1994	4854637.20
INDONESIA	1993	4213131.42
INDONESIA	1992	4999478.51
IRAN	1998	2415763.10
IRAN	1997	4227175.11
IRAN	1996	4527365.03
IRAN	1995	4139514.72
IRAN	1994	4166316.39
IRAN	1993	3366959.59
IRAN	1992	3599399.70
IRAQ	1998	2596922.63
IRAQ	1997	3707054.11
IRAQ	1996	3726138.38
IRAQ	1995	4350503.89
IRAQ	1994	4131512.79
IRAQ	1993	3787196.42
IRAQ	1992	4043738.13
JAPAN	1998	2265666.94
JAPAN	1997	3988819.28
JAPAN	1996	4319004.53
JAPAN	1995	4262698.64
JAPAN	1994	3545212.62
JAPAN	1993	4051565.97
JAPAN	1992	3692137.45
JORDAN	1998	1978591.74
JORDAN	1997	3315454.29
JORDAN	1996	3236531.98
JORDAN	1995	2778207.99
JORDAN	1994	2420301.07
JORDAN	1993	3272130.93
JORDAN	1992	2649126.09
KENYA	1998	2265677.73
KENYA	1997	3493019.32
KENYA	1996	3346373.30
KENYA	1995	3537360.32
KENYA	1994	2800950.72
KENYA	1993	3477468.30
KENYA	1992	2719618.04
MOROCCO	1998	2549499.93
MOROCCO	1997	3891824.90

```
|MOROCCO                   |       1996|            3730777.74|
|MOROCCO                   |       1995|            3469641.13|
|MOROCCO                   |       1994|            3747593.21|
|MOROCCO                   |       1993|            3620742.70|
|MOROCCO                   |       1992|            4303609.25|
|MOZAMBIQUE                |       1998|            2024719.46|
|MOZAMBIQUE                |       1997|            3706003.09|
|MOZAMBIQUE                |       1996|            3376430.93|
|MOZAMBIQUE                |       1995|            2737631.64|
|MOZAMBIQUE                |       1994|            3373146.48|
|MOZAMBIQUE                |       1993|            3608300.37|
|MOZAMBIQUE                |       1992|            3551263.95|
|PERU                      |       1998|            2142791.97|
|PERU                      |       1997|            4664076.15|
|PERU                      |       1996|            3623628.93|
|PERU                      |       1995|            3908939.79|
|PERU                      |       1994|            3386204.16|
|PERU                      |       1993|            3877048.49|
|PERU                      |       1992|            3768394.25|
|ROMANIA                   |       1998|            1760625.70|
|ROMANIA                   |       1997|            2707685.33|
|ROMANIA                   |       1996|            2553345.48|
|ROMANIA                   |       1995|            2715901.59|
|ROMANIA                   |       1994|            3023644.06|
|ROMANIA                   |       1993|            2873247.32|
|ROMANIA                   |       1992|            2728060.71|
|RUSSIA                    |       1998|            2975973.22|
|RUSSIA                    |       1997|            3785806.47|
|RUSSIA                    |       1996|            4217625.59|
|RUSSIA                    |       1995|            3883445.52|
|RUSSIA                    |       1994|            4395855.01|
|RUSSIA                    |       1993|            3900944.18|
|RUSSIA                    |       1992|            4691358.61|
|SAUDI ARABIA              |       1998|            2931482.83|
|SAUDI ARABIA              |       1997|            5498943.16|
|SAUDI ARABIA              |       1996|            4473723.74|
|SAUDI ARABIA              |       1995|            5939212.93|
|SAUDI ARABIA              |       1994|            4527695.71|
|SAUDI ARABIA              |       1993|            4928702.02|
|SAUDI ARABIA              |       1992|            5527261.52|
|UNITED KINGDOM            |       1998|            3198731.37|
|UNITED KINGDOM            |       1997|            4363882.74|
|UNITED KINGDOM            |       1996|            4730956.67|
|UNITED KINGDOM            |       1995|            4842014.55|
|UNITED KINGDOM            |       1994|            4912706.56|
|UNITED KINGDOM            |       1993|            4415255.96|
|UNITED KINGDOM            |       1992|            4375524.23|
|UNITED STATES             |       1998|            1892045.16|
|UNITED STATES             |       1997|            3102027.86|
|UNITED STATES             |       1996|            3334320.26|
|UNITED STATES             |       1995|            3168244.60|
|UNITED STATES             |       1994|            3296960.10|
|UNITED STATES             |       1993|            3558109.05|
|UNITED STATES             |       1992|            2755129.39|
|VIETNAM                   |       1998|            2906627.03|
|VIETNAM                   |       1997|            4544560.45|
|VIETNAM                   |       1996|            4314259.00|
|VIETNAM                   |       1995|            4365340.86|
|VIETNAM                   |       1994|            3686987.71|
|VIETNAM                   |       1993|            3764237.18|
|VIETNAM                   |       1992|            3420922.00|

 Submitting SQL Request #10:

/* 'TPC-D Returned Item Reporting Query (Q10)'                    */

SELECT
  C_CUSTKEY, C_NAME,
  SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT) (FLOAT)) (DECIMAL(18,2)) AS
REVENUE,
  C_ACCTBAL, N_NAME (FORMAT 'x(25)'), C_ADDRESS, C_PHONE (FORMAT
'x(15)'),
  C_COMMENT
  FROM CUSTOMER, ORDERTBL, LINEITEM, NATION
  WHERE
  C_CUSTKEY = O_CUSTKEY
  AND L_ORDERKEY = O_ORDERKEY
  AND O_ORDERDATE >= '1993-10-01'
  AND O_ORDERDATE < ADD_MONTHS('1993-10-01',3)
  AND L_RETURNFLAG = 'R'
  AND C_NATIONKEY = N_NATIONKEY
  GROUP BY C_CUSTKEY, C_NAME, C_ACCTBAL, C_PHONE, N_NAME, C_ADDRESS,
  C_COMMENT
  ORDER BY REVENUE DESC;

 Query 10 complete, 3870 rows returned
 only displaying 20 rows per TPC-D spec

|_______Col1|Col2_____________________|___________________Col3|_________
_____Col4|Col5_____________________|Col6_____________________________
______|Col7___________|Col8_____________________________________________
_____________________________|
|       9722|Customer#000009722       |             464618.26|
474.04|CANADA                   |l Mwzn4NAk6j
|13-518-602-8070|5L 50Oy RSgBAzPxmOSi5wk6xxOR7kh2nnPlgy7LBng2hOw5B01
RmCM120L24Pkg7PS1zwC11BCnz4L6i15PkixP26l66|
|      12800|Customer#000012800       |             444265.64|
1900.84|PERU                     |57zjB3CQx4P4OB2R2MBi2mwhSllM4mn 4 nC6
|27-142-205-3552|0hwglS77RB56Rx436lQ0N16CxhOPnmyhgwz
5z64wnj1kiC4jL350mM41y71hNxBllPjyA4hiN1wzjjM7SCxAN244mk2A|
|       1025|Customer#000001025       |             442028.02|
3363.46|INDIA                    |lkiSn154M5ROi
|18-588-456-4616|0Bl45z233Rniw0OO064nPBgP16kimO0y74iLh73g1N4 m310 jQ
yQzPA50iC 3MA75g2Bj162Nw4P  |
|      13028|Customer#000013028       |             441692.24|
-452.66|UNITED KINGDOM           |yP714ORSNgNN2LA3L5B
|33-253-660-2127|xPkmnhL2BkhkNyww4khlxwwAymN h11PSjBCNMi50LkyOhO6CC
5nzOQCALzliOk2R66w l05hRPO3iSP|
```

```
|       3694|Customer#000003694       |             438180.07|
2960.44|UNITED KINGDOM           |2CCklmCBOCC
|33-421-331-3127|MzLxQxLlLx3MPxlAwg1B5kg61zxkPnk
xiAm6PhMMAAQ2nzN3S6zzgP x70w0lhhPx4QRzlMMy0204lAl3mBO7jh2jAP0N60wg367z|
|        976|Customer#000000976       |             435897.63|
7772.85|ROMANIA                  | QzR 56Px1kgS  wANnAz02RS 30n Pm
|29-436-660-4732|kzn32776 gwzkMzzzO4yxOAnkR7hR4R4x2SMwilz3x6h
nN7OnNLRMml3 kz5SLwi1yklOxiwS4g0wmA5A 4hmgBSwRRiQ1|
|       8206|Customer#000008206       |             429905.11|
6046.36|ARGENTINA                |P yMg30BBBBx NMgC03AmzN2
|11-571-859-1370|hLi122RMPmLC36Oy0kxO71zz2wCR0QQCl7z26hlQ3mM
|
|      13532|Customer#000013532       |             427731.80|
-924.18|KENYA                    |6ij7M5PBMx2kwwyz62Oj4SL5S0mRCwl3m1Rmw
|24-525-332-7244|7ih7yRz214zO67AiNPx64nO5l5k
yj6i3jLA5PCL15Q4QlA3ll60iM1P iBxCixg6 1hCh2RCnjOzk5R OnO 1OhhC3m4631m5|
|      12745|Customer#000012745       |             422327.69|
9691.33|CHINA                    |SgS1LMC4gB2NM3wh
|28-985-189-6174|jl72wjSw0 S6 7L4Cgxw PkyO5Nl2LL7LBR
|
|       2344|Customer#000002344       |             411240.11|
5597.22|MOROCCO                  |O3PC7ikBgw OAzPAlm2P 426zm3BnBN6Q1O
6N  |25-593-745-7663|5NBn0wRNngLw2z5kyn1AhL0ASyg6SMhM
i2kMOyxARAnlO0Q5j4CBNARix7ABlMAC               |
|       2656|Customer#000002656       |             401185.95|
8115.55|ALGERIA                  |On55lAS3Rm5RxS m
|10-667-469-8092|46ABx4jgni mlBMPCLxRhyPQM4RNS 5yO1L7zSOmk
MhPxAxQQ6lQnLj 17LymOhi4l5innzOyB2Olxzmw3gmx0SxiyBN5CSMNgCkLCkMgO|
|         59|Customer#000000059       |             400759.15|
3458.60|ARGENTINA                |wP6CMyClly0lS4CAM1mzm
|11-355-584-3112|l1g7xBCxxC7SM 5AkmnmAk00677O1MzA2R7A0Cx0Njixj56jL2iN
PNkSNQiy55m6ki3OgnhM47mSR7B|
|       7069|Customer#000007069       |             396217.52|
8198.94|INDONESIA                |55Cw7ChL4Bi5ONn2A4m2i2n4nSNQQMjml
|19-644-744-1798|6jNS624l75zlxNli4lxO5zyPykPS1xnilS0NhkgOAkSx7P
|
|       6553|Customer#000006553       |             385863.59|
8985.90|MOZAMBIQUE               |R3LnnxONBjCLC0MRkxy7
|26-166-724-4677|S7CkNLwA3kh006j71lwAlC25Bw6AMQ6i 6C0OSS6O7ARNNny60Ogh
3642mRxyiAgy5yk 3nPO4473wkNg5R6gzO4lz3zmM2m7MiLAilCC|
|       3095|Customer#000003095       |             384246.11|
8829.21|IRAQ                     |S1gMCnBLwzi mCgB664
jl00Ll1Snh1iPMgCgR5 |21-847-218-8188|3LSx7PxS
A4A5Cl3gAy3mg4Qj2xQlyx7xM1kA664AM7zmMmzORh3C1h
MO3nw6MymiljAMg65hOMB4Sn44kO w0lin7|
|       3391|Customer#000003391       |             382541.78|
7742.35|CANADA                   |m3 CORmQNLzkShymLS iMkCimRSl20 NB
|13-592-494-2668|ynMlmhMBA5ikC1nCghlmAhQ0 675S3y2R33yjkNPQOS
|
|      13678|Customer#000013678       |             376280.56|
9030.40|MOROCCO                  |BMk77lQm1lwNA0LghAkg3hCwNl4
|25-306-951-
3937|mOS55RASx1wP136nQ5xBLznLhgw1kQ6PO6imNxQ7kR0x71P0SzByMzh
|
|       6062|Customer#000006062       |             374512.65|
1370.35|CANADA                   |n5zzil60zyxAlkzx7x1nihigPzR OBkR
znMOMh |13-756-700-4918|4zAm4wNB
```

```
li4QRPgPz2wM541x043hmLj4O3LBkALCP16hj2RQBO1OMNly7ww1QP7w5i SSn0jNhAR
yQmmz1hi5j3|
|        554|Customer#000000554       |             373004.47|
8395.57|BRAZIL                   |jC5zhQky4zQB27lB5Sm AQhQ  Px0
|12-938-503-7317|0nxCl3 xSmiLQO 1M
2n0NCiRlnMMxP25j26x2igLhNOxjgMgmwwy7OkjzCACOg0z2LAjOm0RPRmOPiCAAQwLlQSg
1yS3 gLCM1M2BzjnSjPl3nwAkk|
|      13126|Customer#000013126       |             371722.00|
6172.91|INDIA                    | xPAS4MnPh40i5Q2h4NQ61zz4RkyAwANA
|18-288-190-4145|nniMkAN6C0ClQ0mMmPz27liz4hk6L
2MlwPxh42N1l0R2hRwxxzlwMkxO4MAyz7RCj43NxLwQ3m6P27yAj|
```

Submitting SQL Request #11:

```
/* 'TPC-D Important Stock Identification Query (Q11)'             */
```

```
SELECT
  PS_PARTKEY, SUM(PS_SUPPLYCOST * PS_AVAILQTY (FLOAT)) (DEC(18,2)) AS
"VALUE"
  FROM PARTSUPP, SUPPLIER, NATION
  WHERE
  PS_SUPPKEY = S_SUPPKEY
  AND S_NATIONKEY = N_NATIONKEY
  AND N_NAME = 'GERMANY'
  GROUP BY PS_PARTKEY
  HAVING SUM(PS_SUPPLYCOST * PS_AVAILQTY) >
  (SELECT SUM(PS_SUPPLYCOST * PS_AVAILQTY (FLOAT)) * 0.0010000000
  FROM PARTSUPP, SUPPLIER, NATION
  WHERE
  PS_SUPPKEY = S_SUPPKEY
  AND S_NATIONKEY = N_NATIONKEY
  AND N_NAME = 'GERMANY')
  ORDER BY "VALUE" DESC;
```

Query 11 complete, 22 rows returned

Col1	Col2
12098	16227681.21
5134	15709338.52
13334	15023662.41
17052	14351644.20
3452	14070870.14
12552	13332469.18
1084	13170428.29
5797	13038622.72
12633	12892561.61
403	12856217.34
1833	12024581.72
2084	11502875.36
17349	11354213.05

```
|      18427|           11282385.24|
|       2860|           11262529.95|
|      17852|           10934711.93|
|       9871|           10889253.68|
|      12231|           10841131.39|
|       6366|           10759786.81|
|      12146|           10257362.66|
|       5043|           10226395.88|
|      12969|           10125777.93|
```

```
 Submitting SQL Request #12:

/* 'TPC-D Shipping Modes and Order Priority Query (Q12)'          */



SELECT
  L_SHIPMODE (FORMAT 'x(10)'), SUM(CASE WHEN O_ORDERPRIORITY = '1-
URGENT'
  OR O_ORDERPRIORITY = '2-HIGH' THEN 1 ELSE 0 END) AS HIGH_LINE_COUNT,
  SUM(CASE WHEN O_ORDERPRIORITY <> '1-URGENT'
  AND O_ORDERPRIORITY <> '2-HIGH' THEN 1 ELSE 0 END) AS LOW_LINE_COUNT
  FROM ORDERTBL, LINEITEM
  WHERE O_ORDERKEY = L_ORDERKEY
  AND L_SHIPMODE IN ('MAIL', 'SHIP')
  AND L_COMMITDATE < L_RECEIPTDATE
  AND L_SHIPDATE < L_COMMITDATE
  AND L_RECEIPTDATE >= '1994-01-01'
  AND L_RECEIPTDATE < ADD_MONTHS('1994-01-01',12)
  GROUP BY L_SHIPMODE
  ORDER BY L_SHIPMODE;


 Query 12 complete, 2 rows returned


|Col1______|_______Col2|_______Col3|
|MAIL      |        654|        950|
|SHIP      |        684|       1004|
```

```
 Submitting SQL Request #13:

/* 'TPC-D Sales Clerk Performance Query (Q13)'                    */



SELECT
  EXTRACT(YEAR FROM O_ORDERDATE) AS "YEAR",
  SUM(L_EXTENDEDPRICE * (1-L_DISCOUNT) (FLOAT)) (DECIMAL(18,2)) AS
REVENUE
  FROM LINEITEM, ORDERTBL
  WHERE
  O_ORDERKEY = L_ORDERKEY
  AND O_CLERK = 'Clerk#000000088'
  AND L_RETURNFLAG = 'R'
  GROUP BY "YEAR"
  ORDER BY "YEAR";


 Query 13 complete, 4 rows returned


|_______Col1|_____________________Col2|
|       1992|            1262855.73|
|       1993|             964121.03|
|       1994|            1750395.29|
|       1995|             198820.30|
```

```
 Submitting SQL Request #14:

/* 'TPC-D Promotion Effect Query (Q14)'                           */



SELECT
  100.00*SUM(CASE WHEN P_TYPE LIKE 'PROMO%'
  THEN L_EXTENDEDPRICE*(1-L_DISCOUNT) ELSE 0 END (FLOAT))
  / SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT) (FLOAT))
  (DECIMAL (18,2)) AS PROMO_REVENUE
  FROM LINEITEM, PARTTBL
  WHERE
  L_PARTKEY = P_PARTKEY
  AND L_SHIPDATE >= '1995-09-01'
  AND L_SHIPDATE < ADD_MONTHS('1995-09-01',1);


 Query 14 complete, 1 rows returned


|____________________Col1|
|                  16.73|
```

```
 Submitting SQL Request #15:

/* 'TPC-D  Top Supplier Query (Q15)'                              */



CREATE VIEW REVENUE0 (SUPPLIER_NO, TOTAL_REVENUE) AS SELECT
  L_SUPPKEY, SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) (DECIMAL(18,2))
  FROM LINEITEM
```

```
  WHERE
  L_SHIPDATE >= '1996-01-01'
  AND L_SHIPDATE < ADD_MONTHS('1996-01-01',3)
  GROUP BY L_SUPPKEY;


 SQL statement complete.



SELECT
  S_SUPPKEY, S_NAME (FORMAT 'x(25)'), S_ADDRESS, S_PHONE (FORMAT
'x(15)'),
  TOTAL_REVENUE
  FROM SUPPLIER, REVENUE0
  WHERE
  S_SUPPKEY = SUPPLIER_NO
  AND TOTAL_REVENUE = (SELECT
  MAX(TOTAL_REVENUE)
  FROM REVENUE0)
  ORDER BY S_SUPPKEY;


 Query 15 complete, 1 rows returned


|_______Col1|Col2_____________________|Col3_____________________________
_________|Col4___________|__________________Col5|
|        389|Supplier#000000389       |PB1Lx0xx6LMz3h7Rx63m6j3QmMx
|34-885-883-5717|            1418538.21|


DROP VIEW REVENUE0;


 SQL statement complete.



 Submitting SQL Request #16:

/* 'TPC-D Parts/Supplier Relationship Query (Q16)'                */



SELECT
  P_BRAND (FORMAT 'x(10)'), P_TYPE, P_SIZE,
  COUNT(DISTINCT PS_SUPPKEY) AS SUPPLIER_CNT
  FROM PARTSUPP, PARTTBL
  WHERE
  P_PARTKEY = PS_PARTKEY
  AND P_BRAND <> 'Brand#45'
  AND P_TYPE NOT LIKE 'MEDIUM POLISHED%'
  AND P_SIZE IN (49,14,23,45,19,3,36,9)
  AND PS_SUPPKEY NOT IN (SELECT S_SUPPKEY FROM SUPPLIER
  WHERE S_COMMENT LIKE '%Better Business Bureau%Complaints%')
  GROUP BY P_BRAND, P_TYPE, P_SIZE
  ORDER BY SUPPLIER_CNT DESC, P_BRAND, P_TYPE, P_SIZE;


 Query 16 complete, 2762 rows returned, 20 rows displayed


|Col1_____|Col2_____________________|_______Col3|_______Col4|
|Brand#14 |SMALL ANODIZED NICKEL    |         45|         12|
|Brand#22 |SMALL BURNISHED BRASS    |         19|         12|
|Brand#25 |PROMO POLISHED COPPER    |         14|         12|
|Brand#35 |LARGE ANODIZED STEEL     |         45|         12|
|Brand#35 |PROMO BRUSHED COPPER     |          9|         12|
|Brand#51 |ECONOMY ANODIZED STEEL   |          9|         12|
|Brand#53 |LARGE BRUSHED NICKEL     |         45|         12|
|Brand#11 |ECONOMY POLISHED COPPER  |         14|          8|
|Brand#11 |LARGE PLATED STEEL       |         23|          8|
|Brand#11 |PROMO POLISHED STEEL     |         23|          8|
|Brand#11 |STANDARD ANODIZED COPPER |          9|          8|
|Brand#12 |ECONOMY BURNISHED BRASS  |          9|          8|
|Brand#12 |LARGE ANODIZED BRASS     |         14|          8|
|Brand#12 |SMALL ANODIZED TIN       |         23|          8|
|Brand#12 |SMALL BRUSHED NICKEL     |         23|          8|
|Brand#12 |STANDARD ANODIZED BRASS  |          3|          8|
|Brand#12 |STANDARD BURNISHED TIN   |         23|          8|
|Brand#13 |ECONOMY POLISHED BRASS   |          9|          8|
|Brand#13 |LARGE BURNISHED COPPER   |         45|          8|
|Brand#13 |MEDIUM ANODIZED STEEL    |         23|          8|


 Submitting SQL Request #17:

/* 'TPC-D Small-Quantity-Order Revenue Query (Q17)'               */



SELECT
  SUM(L_EXTENDEDPRICE) / 7.0 AS AVG_YEARLY
  FROM LINEITEM, PARTTBL
  WHERE
  P_PARTKEY = L_PARTKEY
  AND P_BRAND = 'Brand#23'
  AND P_CONTAINER = 'MED BOX'
  AND L_QUANTITY < (SELECT
  0.2 * AVG(L_QUANTITY)
  FROM LINEITEM
  WHERE L_PARTKEY = P_PARTKEY);


 Query 17 complete, 1 rows returned


|_______________Col1|
|           24436.88|
```

Appendix D. Query Substitution Parameters and Seeds Used

```
STREAM 00   seed = 68870749

1       69
2       9       STEEL   ASIA
3       AUTOMOBILE      1995-03-30
4       1993-11-01
5       AFRICA 1997-01-01
6       1993-01-01      0.09    25
7       EGYPT   UNITED KINGDOM
8       EGYPT   MIDDLE EAST     PROMO POLISHED STEEL
9       chiffon
10      1993-05-01
11      EGYPT   0.0000010000
12      AIR     SHIP    1995-01-01
13      Clerk#000000163
14      1993-11-01
15      1993-11-01
16      Brand#15        LARGE ANODIZED 42 16 30 39 8 43 4 3
17      Brand#15        MED PACK

STREAM 01   seed = 68870750

1       83
2       20      TIN     AFRICA
3       BUILDING        1995-03-01
4       1994-12-01
5       AMERICA 1993-01-01
6       1994-01-01      0.02    24
7       INDONESIA       ALGERIA
8       INDONESIA       ASIA    STANDARD ANODIZED TIN
9       honeydew
10      1993-11-01
11      INDONESIA       0.0000010000
12      RAIL    REG AIR 1993-01-01
13      Clerk#000000382
14      1994-12-01
15      1994-12-01
16      Brand#21        SMALL ANODIZED 39 36 6 31 10 46 30 23
17      Brand#21        SM DRUM

STREAM 02   seed = 68870751

1       96
2       31      BRASS   MIDDLE EAST
3       HOUSEHOLD       1995-03-02
4       1995-12-01
5       EUROPE  1993-01-01
6       1996-01-01      0.02    25
7       MOROCCO ARGENTINA
8       MOROCCO AFRICA  STANDARD BURNISHED BRASS
9       navajo
10      1994-04-01
11      MOROCCO 0.0000010000
12      MAIL    REG AIR 1997-01-01
13      Clerk#000000601
14      1996-02-01
15      1995-12-01
16      Brand#41        PROMO BURNISHED 35 5 23  11  48  6  44  41
17      Brand#41        WRAP JAR

STREAM 03   seed = 68870752

1       110
2       41      COPPER  ASIA
3       MACHINERY       1995-03-03
4       1997-01-01
5       MIDDLE EAST     1993-01-01
6       1997-01-01      0.02    25
7       SAUDI ARABIA    BRAZIL
8       SAUDI ARABIA    MIDDLE EAST     STANDARD PLATED COPPER
9       sandy
10      1994-09-01
11      SAUDI ARABIA    0.0000010000
12      FOB     REG AIR 1995-01-01
13      Clerk#000000820
14      1997-03-01
15      1997-01-01
16      Brand#51        LARGE BURNISHED 32 24 8  14  13  1  33  40
17      Brand#51        MED CAN
```

Appendix E. Database Content of Initial Ten Rows

Lineitem Table

```
sel * from lineitem where l_orderkey in (1,2,3) order by
1,l_linenumber;

 *** Query completed. 13 rows found. 16 columns returned.

 L_ORDERKEY    L_PARTKEY    L_SUPPKEY  L_LINENUMBER         L_QUANTITY
-----------  -----------  -----------  ------------  -----------------
     L_EXTENDEDPRICE         L_DISCOUNT              L_TAX
L_RETURNFLAG
   -----------------  -----------------  -----------------  -----------
-
   L_LINESTATUS  L_SHIPDATE  L_COMMITDATE  L_RECEIPTDATE
   ------------  ----------  ------------  -------------
   L_SHIPINSTRUCT             L_SHIPMODE
   -------------------------  ----------
   L_COMMENT
   ------------------------------------------
          1     15518935       768951             1              17.00
            33203.72               .04                .02  N
   O             1996-03-13    1996-02-12     1996-03-22
   DELIVER IN PERSON          TRUCK
   iPBw4mMm7w7kQ zNPL i261OPP
          1      6730908       730909             2              36.00
            69788.52               .09                .06  N
   O             1996-04-12    1996-02-28     1996-04-20
   TAKE BACK RETURN           MAIL
   5wM04SNyl0AnghCP2nx lAi
          1      6369978       369979             3               8.00
            16381.28               .10                .02  N
   O             1996-01-29    1996-03-05     1996-01-31
   TAKE BACK RETURN           REG AIR
   SQC2C 5PNCy4mM
          1       213150       463151             4              28.00
            29767.92               .09                .06  N
   O             1996-04-21    1996-03-30     1996-05-16
   NONE                       AIR
   Om0L65CSAwSj5k6k
          1      2402664       152671             5              24.00
            37596.96               .10                .04  N
   O             1996-03-30    1996-03-14     1996-04-01
   NONE                       FOB
   C2gOQj OB6RLk1BS15 igN
          1      1563445        63448             6              32.00
            48267.84               .07                .02  N
   O             1996-01-30    1996-02-07     1996-02-03
   DELIVER IN PERSON          MAIL
   CB0SnyOL PQ32B70wB75k 6Aw10m0wh
          2     16818625       818626             1              24.00
            37026.72               .00                .08  N
   O             1997-03-05    1997-02-09     1997-03-11
   COLLECT COD                AIR
   O52M70MRgRNnmm476mNm
          3     10616973       116994             1              38.00
            71798.72               .00                .05  A
   F             1993-12-11    1993-11-27     1993-12-16
   TAKE BACK RETURN           RAIL
   3AR yMS77lQ12kR
          3     19450826       700846             2              30.00
            53275.50               .05                .01  A
   F             1994-02-05    1993-12-29     1994-02-18
   TAKE BACK RETURN           FOB
   6wQnO0Llg6y
          3     10016338        16339             3              44.00
            55168.52               .07                .03  R
   F             1994-01-19    1993-12-23     1994-02-15
   TAKE BACK RETURN           SHIP
   LhiA7wygz0k4g4zRhMLBAM
```

Order Table

```
sel * from ordertbl where o_orderkey in (1,2,3,4,5,6,7,32,33,34) order
by 1;

 *** Query completed. 10 rows found. 9 columns returned.

 O_ORDERKEY    O_CUSTKEY  O_ORDERSTATUS       O_TOTALPRICE
-----------  -----------  -------------  -----------------
   O_ORDERDATE
   -----------
   O_ORDERPRIORITY  O_CLERK          O_SHIPPRIORITY
   ---------------  ---------------  --------------
   O_COMMENT
   -------------------------------------------------------------------------
-----------
          1      3689999  O                     224560.83
    1996-01-02
   5-LOW            Clerk#000095055               0
   A0xCm5ARNL mxjChn2kC64xA4L6zBg2O5jhg M42izyPO  QlymN1ky5kmSiSgBAQA
          2      7800163  O                      39988.85
    1996-12-01
   1-URGENT         Clerk#000087916               0
   5PRxL1nM7xhQNzP2hnjhy1zz ykhg4P2A MMg5Px3OCN 0B0iyCRgiC2
          3     12331391  F                     306426.43
    1993-10-14
   5-LOW            Clerk#000095426               0
   nm0kygQBnw7RS3AAA4k
          4     13677602  O                      7884.05
    1995-10-11
   5-LOW            Clerk#000012340               0
   CP42CySQlz64n3mCyjm17 4B0CL
L5772m4k2Ai4h1nPySwSmNyCl4jOAOx5y4Rjx36nhO1x2x4Qw
          5      5562202  F                     184997.17
```

```
    1994-07-30
   5-LOW            Clerk#000092480              0
    3PNC7zMP534MSizgy34Bxj62l0C7n6PBk7
          6      3913430  F                     4195.11
    1992-02-21
   4-NOT SPECIFIED  Clerk#000005798              0
   1CN00NA0z75SwwCxMNB0MLNL
          7     13005694  O                    205492.90
    1996-01-10
   2-HIGH           Clerk#000046961              0
   gmiC6hj5L4 0ixCAQkmB6giC16l4L16g
         32      6695788  O                    214045.34
    1995-07-16
   2-HIGH           Clerk#000061561              0
   7ihNSz00NCxA31PPx6RM4ih BPPlz417SLk3SRA1zx0nlikRgjkx
         33      6100004  F                    107296.91
    1993-10-27
   3-MEDIUM         Clerk#000040860              0
   jkACLh 0igMiy72n Sky0h0B6NB70j7Q
         34      8611441  O                    128003.89
    1998-07-21
   3-MEDIUM         Clerk#000022278              0
   05k 2x242klm jyA wB0CBzzQnz5P11nAml5AL5jC lg5


sel * from parttbl where p_partkey in (1,2,3,4,5,6,7,8,9,10) order by
1;

 *** Query completed. 10 rows found. 9 columns returned.

  P_PARTKEY  P_NAME
-----------  ----------------------------------------------------------
   P_MFGR                    P_BRAND
   ------------------------  ----------
   P_TYPE
   -------------------------
        P_SIZE
   -----------
   P_CONTAINER      P_RETAILPRICE
   -----------  -----------------
   P_COMMENT
   ----------------------
          1  goldenrod lace spring chartreuse ivory
   Manufacturer#1            Brand#13
   PROMO BURNISHED COPPER
            7
   JUMBO PKG               901.00
   zMg1PACmQ 7RCCC7
          2  snow ghost azure burnished lemon
   Manufacturer#1            Brand#13
   LARGE BRUSHED BRASS
            1
   LG CASE                 902.00
   Bxg4RlO6051n7NjN zn
          3  cornflower navajo salmon lemon orchid
   Manufacturer#4            Brand#42
   STANDARD POLISHED BRASS
           21
   WRAP CASE               903.00
   4241RR3By
          4  olive dim lemon light khaki
   Manufacturer#3            Brand#34
   SMALL PLATED BRASS
           14
   MED DRUM                904.00
   z1n7znz6
          5  lavender cornsilk linen seashell lemon
   Manufacturer#3            Brand#32
   STANDARD POLISHED TIN
           15
   SM PKG                  905.00
   gj4Lg5BhBk12iS
          6  cornsilk beige chartreuse medium blue
   Manufacturer#2            Brand#24
   PROMO PLATED STEEL
            4
   MED BAG                 906.00
   yNjzS Njyh4mgLx Om
          7  honeydew purple cream mint coral
   Manufacturer#1            Brand#11
   SMALL PLATED COPPER
           45
   SM BAG                  907.00
   PSNg0L
          8  puff blush tomato papaya navy
   Manufacturer#4            Brand#44
   PROMO BURNISHED TIN
           41
   LG DRUM                 908.00
   k042AL4y21N1yNPC77
          9  burnished violet pink rose drab
   Manufacturer#4            Brand#43
   SMALL BURNISHED STEEL
           12
   WRAP CASE               909.00
   37PLkwhgiAP0xCkxO
         10  slate dark white lavender purple
   Manufacturer#5            Brand#54
   LARGE BURNISHED STEEL
           44
   LG CAN                  910.01
   wPP74M1Lwj1
```

Partsupp Table

```
sel * from partsupp where ps_partkey in (1,2,3) order by 1,ps_suppkey;

 *** Query completed. 12 rows found. 5 columns returned.

 PS_PARTKEY  PS_SUPPKEY
-----------  -----------
  PS_AVAILQTY     PS_SUPPLYCOST
  -----------  -----------------
  PS_COMMENT
```

```
  ----------------------------------------------------------------
----------------------------------------------------------------
----------------------------------------------------------
          1           2
         3325            771.64
  00PL56QkQRSkg2z7MANNj4i1h2zLQQLiQnAlML1S6 k4hg3hP5hk3ywMLwy 7gjR3
4Q7S1Qmzx2jOS37Mk61n yCg4Q7k522P0055wg23B0Mw3BOwSy6z5Q6xljABx3LAj6R6CmM
l4jjMzQ02LkiiyCCwBk7w465kLBz7QlCk26ARLOxk7z2hC0jw7
          1      250002
         8076            993.49
  nSO7Mln4N7LlxgAyM2MzNnO7k0NlhjyShgCy30A
27QML0SQ77CPPgkCQAQCwz5M3MmSSAQ LxMLMC0Bj4CNN276SmQRSljPxz5z3
L2mLMQSBghjLnCOR4Nl 156OMP C76QL
xiyw0kSQy1w6ygAxOA4hx7Ngihi5NAPNl2LQ4SRnNhn7mygOB0z
          1      500002
         3956            337.09
  6215k
jLCizNlOBl62nP4lLQy431kOzyzn2M6L3h73lCljhlx3x5ghjlOyL76A0hzPk2CS2jkxN
gAN3gnk652 Cj4k4
          1      750002
         4069            357.84
  234OCA5ghw0P0gS3n2jCS35yAm 3L5C7iB k7 w1 R52LLOACQ6i606OB
2MP1x0wC23ik2Omk4NnxzmS6z5z5il661l2l2g0P3OLk66jzQxjSSAnwSnQ3xz
QBQR2j10hNmmyQ14hl45l4x5C B5Qz Lk26yhQNmS54A2O7wlC P0C57CyxSL3
          2           3
         8895            378.49
  MMNOM3BnMM6NBzjB 2mg i jALB
nQhBM5ROi5N7A5w4B4S2k1506OzMgh6SRB7nP1hQCjgjR17SBA77g6niCwi0L6Pghh1S0O4
mlSL1OShRkyxQS7NNQj
          2      250003
         4969            915.27
  6S66zNlykhii26wwAxz1PRMxggAy446yylPBS5wP
Rl6ggNkyikkyhxymMShNgQmBim1N60
00NSjwPw02lhPPSmn3yRSCn105llnPBk2MlRlxxQAmR mO2kxiRh5Pk4x2OnS40nnQRm
16L6NC2RSAkL36g6w64L5w4w74Lnz5wROSOQNingx4imSnPwz5N hSxNg
          2      500003
         8539            438.37
  BPOgj3k MgQR2
x6kn3BR6lkmhMzjQk6S343LmN6lz2Qh512MSi4nQ5Bghxlh401yyn463mxOh7Bxgx1PR60j
wSMQ661OCA6mPO33R05R2S6N330in0Qx0AC4QhBnOkz4NlQkMR2gh3 k1xk
SjN40C77hm2Qlh01jAkSP
          2      750003
         3025            306.39
  y5BNy3Aw02nxyMxgzP5BS14gg7MCnLlSkih56gOMOy4QyNj5P3iM
jOmkRQhRR3hylCn4jN LlgSxyPigjRLgBygM
RR4CL3Pjx6CRMNl1iA7w2ALwkzn06khOOzyQNB63wP4BLSk wPNk4My
          3           4
         4651            920.92
  P7 437MmnM0Pik
lAwBj0gSnm1z1zAMA6417zgS154nLCC0Q6BC1lgxyB6BkOj6QCC6n4mw2w7jgCNP
zz5AMw37 z
          3      250004
         4093            498.13

PyRmlwO76kO3igxhS64h5x6PBLM2PxkO0j3NMRgzP6S1ghhw4Nnn04my03lzCyQQ4M3gS2k
O2iOmxPzlCM330zN1yPnzml1ixgB
```

```
sel * from customer where c_custkey  in (1,2,3,4,5,6,7,8,9,10) order by
1;

 *** Query completed. 10 rows found. 8 columns returned.

  C_CUSTKEY  C_NAME
-----------  -------------------------
   C_ADDRESS
   ----------------------------------------
   C_NATIONKEY
   -----------
   C_PHONE
   ---------------
           C_ACCTBAL
   -----------------
   C_MKTSEGMENT
   ------------
   C_COMMENT
   ----------------------------------------------------------------
-------------------------------------------------
          1  Customer#000000001
   ANhzAAh6R3 glS4Sx
             15
   25-989-741-2988
              711.56
   BUILDING
   j5S37kk6zkOzkM5NOz6jwwimkN66CmOhM5ySy w6PAj2xjOAmhkw6ChSR
21BMRkL0kLM5zxg654CRlB3 1Lxm3S
          2  Customer#000000002
   MN0L3OzNgylx2
             13
   23-768-687-3665
              121.65
   AUTOMOBILE
   M4QB23ixkg0yk6m3gwim6zi32PS7lj2
          3  Customer#000000003
   PSL74SNCwwN2ON66lxgnw7mR4hLP2k
             1
   11-719-748-3364
              7498.12
   AUTOMOBILE
   mSC13MBj4n0P6Mgh0ml02zOBlyjw3NzB1
          4  Customer#000000004
   mknn1Sh0NPMz1k5Lw2OB mO
             4
   14-128-190-5944
              2866.83
   MACHINERY
   MN6ChhSMwPwzOkyww7C5ROlhMS0C4iR2nC6kQmywx3yim62QNySOmQRQnwizihMOg
          5  Customer#000000005
   yOww5znhPNi5OlQNPChkLx2BLPxNSB
             3
   13-750-942-6364
              794.47
   HOUSEHOLD
   24BOSzg 03m710wll   iNxnwQ0OmzzgO7A3ykBj2
g755hhCyMO7QnARx5Pg3kyAQA35 i0CS1MSLg0xN2iyg0liwnMwnOx52nj5iQkNQPP
          6  Customer#000000006
```

```
   nS70ykL4n k51ik3R5wlNzjnjBL2N51ki
            20
   30-114-968-4951
             7638.57
   AUTOMOBILE
   hPMLmxPw05R1mz126jjRAj1kOP7xLC6
yS3ALCRBR5B3im650BLm4O3SwBP7xlwOklmPRS31RNN0gMkkPm4COigCRMlniz27jwg63yz
          7  Customer#000000007
    ChljB04OgAizN6kQhRi7LjjNiCM0A AS
            18
   28-190-982-9759
             9561.95
   AUTOMOBILE
   QM63L2miSw3hy34iQ11235 011mkgk0SkCRC73L1CgiLROzNwjO4PQSBx2n2iQg5h
          8  Customer#000000008
   kCRz0CknMw7mh4P50QjBnxSLRxQCMOAh yNn
            17
   27-147-574-9335
             6819.74
   BUILDING
   x1Rh1P5M73Lix xyM Lmng0RO4MBQyLll7wzwgyOLCxi2yCLgL1zO4yOiAPj
          9  Customer#000000009
   L4z65g2RNgN6PxM5kRjnPB7k2kwL62
             8
   18-338-906-3675
             8324.07
   FURNITURE
   7zRiSzmj4Ak7L6N7RljhM5437B6CPmP54RC1x1x7C6hziN6l
         10  Customer#000000010
   L3jg3xAwi6A0B103B0Aymm
             5
   15-741-346-9870
             2753.54
   HOUSEHOLD
   7Lm LiCwwxQMykgNOR6kzCyP1B21QyA57hBlSOPnx6m53iSOP6w44M3CP
MnP7Alky4OwkOwSh20341
```

Supplier Table

```
sel * from supplier where s_suppkey  in (1,2,3,4,5,6,7,8,9,10) order by
1,2;

 *** Query completed. 10 rows found. 7 columns returned.

  S_SUPPKEY  S_NAME
-----------  -------------------------
   S_ADDRESS
   ----------------------------------------
   S_NATIONKEY
   -----------
   S_PHONE
   ---------------
           S_ACCTBAL
   -----------------
   S_COMMENT
   ----------------------------------------------------------------------
---------------------------------
          1  Supplier#000000001
    N kw4gn1OM Ahw3Sg70BBgQw57lgjzj55R
             17
   27-918-335-1736
              5755.94
   lLniMi51QPmO1 C2hy27wkN21mmg53 BhQBB1O2x4OmiR4kO5kN1BS 4PwMhk
Pk2nRnA2 k
          2  Supplier#000000002
   ji3yh016B5
              5
   15-679-861-2259
              4032.68
   B32z0yzh2lPyOwQkAjA704yM2R7llRlk2
xCliy41QNmQn0RNl0OQ4jgMy3kSRBLzyw25CB5 lk0A 54
          3  Supplier#000000003
   mxBQBnxO3CSwl7
              1
   11-383-516-1199
              4192.40
   BS0OzjiO1yM6Rgl4mxLNhjSMPB37Sw7ym3R7ll2n4SSCilz6nlL5SBOig
          4  Supplier#000000004
   7zR323R73NMB77wi1
             15
   25-843-787-7479
              4641.08
   w lQn6QyOSSxhw10C6gz2BngiLRAMmgnRxiLiO3
          5  Supplier#000000005
   AmMQ7Mg 10ByLCP52M13xN31jh5hzOgnm00B
             11
   21-151-690-3663
              -283.84
   PAziBQQixjwS7P4lQhn10i74050M
AzkxACnOAyjnSm3CQ26SOx5kynSR0nlLSzi3y3nzPPNikNl3P3 kLwwOP7AM30CO0ymAh
          6  Supplier#000000006
   QQL6hxmnMkkgMwgm7CB5B 30Llz
             14
   24-696-997-4969
              1365.79
   giSki24 gRNAmB 1yOzPR6Q2kiNCQ0h3LLyxmROA5O7OOi5zlzy
          7  Supplier#000000007
   z45m2jBRzl5ilLNz4
             23
   33-990-965-2201
              6820.35
   1PhngjmiSQl0RzRACP0l4S70xSL
QPSBM16072SkMLCgm4OOMjARLNQk3g1P3BB32AgBMl462B0CP7Rh24
          8  Supplier#000000008
   xz5m4C  A4AAj0kANQ
             17
   27-498-742-3860
              7627.85
   lz57Mw6RNwCSCzmAShwN7S45w20C5zS6zi 5Al1ORMwnQmjS5SgBnRhQ1lCkyBlhN
6MP7 kAzNw3gSjyyLMiNzhCmPn0 g5x23Q
          9  Supplier#000000009
   m7k7CnC3wiP
             10
   20-403-398-8662
              5302.37
   xPLzNgk5nzA jm3PLmySlm PS  zANRjSgh2njAg
         10  Supplier#000000010
```

```
   wN1S4mQ0g7Px5Lj34xw6kS4Li4NzB4mO
            24
   34-852-489-8585
               3891.91
   5xwg6AOz0NzhONL6kC43zR3AhzO6njCiwPg7k6MxwP1mN2 Rg 5Q426
```

Nation Table

```
sel * from nation where n_nationkey in (0,1,2,3,4,5,6,7,8,9) order by
1;

 *** Query completed. 10 rows found. 4 columns returned.

N_NATIONKEY  N_NAME
-----------  -------------------------
   N_REGIONKEY
   -----------
   N_COMMENT
   ------------------------------------------------------------------------
------------------------------------------------------------------------
-------------
           0  ALGERIA
              0
   2Cxhl7 L1iwk6hMh300izngN32CPwCikyLk6khMzSRA
           1  ARGENTINA
              1
   zQn3Okwz1wLn7PLS3OhCgn56kP5PyRikgi1B7lL
           2  BRAZIL
              1
   gLmS0nACAmnBCj2klki7RCPNgPxnCOjNg4k OiAg57COSOm1NwCnOyLx40R SC
y20gPPAkNk5hxRhR5OmgS1iPQQzNAxPL30n67OgyC l617Sh4LS
           3  CANADA
              1
    4yMO AhnQ5Lh wzQAM662Aw1ByCl7CxmzRwNR5nAlO4 x
           4  EGYPT
              4
   l1im5126  Cxj NMQmLxOikni02j2m3Ah4yNR1QQiL507j2QSlyN
           5  ETHIOPIA
              0
   NS7n LSOP Oz5n1AlB2S02nN0lMh4SBxP iRhBO 047R26 2BlM
           6  FRANCE
              3
   3mjmizl S 3L3k2hNNhNlP4w370xRxyN15wn
           7  GERMANY
              3
   z nOP4RkwO CmzBB 516mAg lByw4OM3QyNPA
           8  INDIA
              2
   MNlR5RCiRMj1l11wjN7Myn M1lylNlMmBQl7PL4C
kkxQkgPQ7i3w6B67R2QkOO40xl4Q2iw76jRL7ilhR5Q 0xC7RRm5iQ2NAx2LiBm3QiO27j
           9  INDONESIA
              2
   SjPmQO71Lj 7ABj6MxlAQk3nLwi73BPxzCwjzMn4zlLzgg6nnz0j0w
zxC66gP6ykRPMg
```

Region Table

```
sel * from region where r_regionkey in (0,1,2,3,4) order by 1;

 *** Query completed. 5 rows found. 3 columns returned.

R_REGIONKEY  R_NAME
-----------  -------------------------
   R_COMMENT
   ------------------------------------------------------------------------
------------------------------------------------------------------------
-------------
           0  AFRICA
   xSx31zz31Cl1z4OAnmm05AjiOxC3AMMNOgC0kACgwngg3glP7LLLywlQy7R
           1  AMERICA

kgyh3LSnC72k6zlAz0LP3k2L4QB1QL1O673OjO1SPj0ngQ7CO100SBgmgRQ4lgPCMk21A42
5iklyAR4yBRAwR4Cm5miNw 4jl13mMnxw17B
           2  ASIA
   NSg6xlMlA1lzm6mOR0Ajx nhRA77NgRxBwL1M6Py RjySB3RLwkyPkwMM2R1BQ
xAzkOgkjmll0gAghinP5inmNmR76MlijMS3S2zxONR15
           3  EUROPE

zlSL7Qwg12hMBL5lhlz0M45QkjShwSyiO04MLOh7wn1ARLQPyPAyAiil576l1Li7AlnR1S
RQ4SLny7B2Ryj5P66MLhn NxhwB4C3ig0SO
           4  MIDDLE EAST
   RllxmhPLz3Cy2mNlg4QMBnNASM ACki MPki7Oi
```

Appendix F. Data Load Components

Description of the loading method

For this TPC-D benchmark, two methods of loading data were employed. The first method uses a freely available parallel load utility called PFAST combined with a version of the DBGEN data generator redesigned to be a subroutine called by PFAST for each row to be loaded. Such subroutines are referred to as “Inmod” routines. The PFAST utility uses standard Teradata call-level interface (CLIv2) to communicate the data to the SUT.

The second method involves using the standard version of DBGEN in conjunction with it's in line load option (-d). We replaced the stub in line load routines that came with DBGEN with a routine that automatically invokes the standard Teradata load utility FASTLOAD, and uses a named pipe to transfer the data as generated by DBGEN to FASTLOAD. FASTLOAD then uses the standard Teradata call-level interface (CLIv2) to communcate the data to the SUT.

While either method can be used to load any or all of the TPC-D database, the second method is easy to use and easy to understand, while the first method is somewhat faster. Because of this, the first method was used for all tables, with the exception of Nation and Region, while the second method was used for the remaining tables.

F.1 Inmod for FastLoad used by Pfast

The Inmod subroutine is made up of a number of pieces. The file blkexit.c is the main body of the subroutine, and is a modification of the driver.c code from the standard DBGEN. Tdinload.c is a set of utility routines used by blkexit.c. The files build.c, bm_utils.c, rnd.c, print.c, bcd2.c, and speed_seed.c are used as-is from the standard DBGEN distribution with no modifications. All header files from DBGEN are used as-is from the standard DBGEN distribution with no modifications.

Makefile

```
#  Sccsid:     @(#)makefile.suite     9.1.1.10     8/28/95  10:21:42
# makefile for DSS benchamrk data generator
#
#
#################
## CHANGE NAME OF ANSI COMPILER HERE
#################
SRCNODE=bynet552
SRCTREE=/home2/ac4/dbgen_pfast
DSTTREE=/home2/ac4/pfast/inmod
DOALL=/home2/fpf/bin/doall

CC      = cc
LDFLAGS =
LIBS    = -lcliv2 -lsocket -lnsl  -lm -lc -lgen
CFLAGS  = -O -DTDAT -DPFAST -DSTDLIB_HAS_GETOPT -D__STDC__ -o6 -586 -
I/usr/include -I/usr/ucbinclude
#
# NO CHANGES SHOULD BE NECESSARY BELOW THIS LINE
################

POBJ = lineitem.o order.o part.o partsupp.o supplier.o customer.o
PROG  = ${POBJ:.o=}

HDR = dss.h rnd.h config.h dsstypes.h shared.h bcd2.h
SRC = build.c blkexit.c bm_utils.c rnd.c print.c tdinload.c bcd2.c
speed_seed.c
OBJ = build.o bm_utils.o rnd.o print.o tdinload.o bcd2.o speed_seed.o
SLAVE=SLAVE/fastslav.o
SETS = dists.dss /usr/lib/libcliv2.a /usr/ucblib/libucb.a -lsocket -
lnsl


all: $(PROG)

$(PROG): $$@.o $(OBJ) ${SLAVE}
        $(CC) $(LDFLAGS) -o $@ $@.o $(OBJ) ${SLAVE} $(LIBS)

lineitem.o: blkexit.c dss.h dsstypes.h config.h makefile
        $(CC) $(CFLAGS)  -O -DTABLE=LINE -o $@ -c blkexit.c

order.o: blkexit.c dss.h dsstypes.h config.h makefile
        $(CC) $(CFLAGS) -O -DTABLE=ORDER -o $@ -c blkexit.c

part.o: blkexit.c dss.h dsstypes.h config.h makefile
        $(CC) $(CFLAGS) -O -DTABLE=PART -o $@ -c blkexit.c

partsupp.o: blkexit.c dss.h dsstypes.h config.h makefile
        $(CC) $(CFLAGS)  -O -DTABLE=PSUPP -o $@ -c blkexit.c

supplier.o: blkexit.c dss.h dsstypes.h config.h makefile
        $(CC) $(CFLAGS) -O -DTABLE=SUPP -o $@ -c blkexit.c

customer.o: blkexit.c dss.h dsstypes.h config.h makefile
        $(CC) $(CFLAGS) -O -DTABLE=CUST -o $@ -c blkexit.c

${SLAVE}:
        @echo Updating pfast slave;
        cd SLAVE; $(MAKE)
        @echo done;

distribution: all
        @echo distributing lineitem inmod
        $(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/lineitem
${DSTTREE}/lineitem
        $(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/lineitem
${DSTTREE}/lineitem1
        $(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/lineitem
${DSTTREE}/lineitem2
        $(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/lineitem
${DSTTREE}/lineitem3
        @echo distributing order inmod
        $(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/order ${DSTTREE}/order
        $(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/order ${DSTTREE}/order1
        $(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/order ${DSTTREE}/order2
        $(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/order ${DSTTREE}/order3
        @echo distributing part inmod
```

```
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/part ${DSTTREE}/part
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/part ${DSTTREE}/part1
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/part ${DSTTREE}/part2
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/part ${DSTTREE}/part3
	@echo distributing partsupp inmod
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/partsupp
${DSTTREE}/partsupp
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/partsupp
${DSTTREE}/partsupp1
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/partsupp
${DSTTREE}/partsupp2
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/partsupp
${DSTTREE}/partsupp3
	@echo distributing supplier inmod
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/supplier
${DSTTREE}/supplier
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/supplier
${DSTTREE}/supplier1
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/supplier
${DSTTREE}/supplier2
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/supplier
${DSTTREE}/supplier3
	@echo distributing customer inmod
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/customer
${DSTTREE}/customer
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/customer
${DSTTREE}/customer1
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/customer
${DSTTREE}/customer2
	$(DOALL) rcp -p ${SRCNODE}:${SRCTREE}/customer
${DSTTREE}/customer3

clean:
	rm -f $(PROG)  $(OBJ)  ${SLAVE}


rnd.o: rnd.h
$(OBJ): config.h makefile
$(OBJ): dss.h dsstypes.h
```

blkexit.c

```
/*************************************************************
This is a Teradata-specific version of the DBGEN program for TPC-D.
This file replaces driver.c from the original DBGEN program.  This
file is used in conjunction with tdatload.c, which replaces the file
load_stub.c.  load_stub.c is suppossed to be customized for each DB.

Note that there are NO teradata specific changes
made to any of the header (.h) or other source (.c) files, including
those that actually generate the data (such as build.c).

Replacing driver.c is necessary because rather than a main program,
we want this routine to be a subroutine that can be called by the
Teradata load utility (FASTLOAD).  Such routines are called "INMOD"s.
Although we a replacing driver.c, much of the driver.c source is
embedded in this file unchanged.  I've marked what parts are copied
straight from driver.c, and which parts are Teradata specific changes.

Since an FASTLOAD can only load one table per run, this INMOD needs
to know which file to generate rows for.  This is handled by a
define called "TABLE".  It must be defined to be one of the table
constants PART,PSUPP,SUPP,CUST,ORDER,LINE,TIME,NATION,REGION.
It can be defined as a compiler option -DTABLE=CUST, or it defaults
to LINE via the define just below this comment.

Also, because we have a special version of FASTLOAD for loading
large amounts of data (called PFAST) that requires slightly different
INMODs, you can define PFAST before compiling to get an inmod designed
for PFAST. Note that PFAST has a strange restriction:  INMODs must
NOT write to stdout.  So I changed all printfs to go to stderr.

Because we aren't a main program, we can't expect command-line
arguments, so we read the information we need from a file called
"inmod.info" in the current directory.  This file needs to contain
three numbers:  The scale factor, the total number of child processes
being run, and which step we are generating for.  These are the same
as the command line arguments -s, -C, and -S.
If the step is 0 or -1, we generate the entire table.
*************************************************************/
#ifndef TABLE
#define TABLE LINE
#endif

/*** The following code is from driver.c with no Teradata
modifications: */
/* Sccsid:     @(#)driver.c 9.1.1.30     9/7/95  16:24:03 */
/* main driver for dss banchmark */

#define DECLARER        /* EXTERN references get defined here */
#define NO_FUNC (int (*) ()) NULL   /* to clean up tdefs */
#define NO_LFUNC (long (*) ()) NULL   /* to clean up tdefs */
#define LIFENOISE(cnt) if (i % cnt == 0 && verbose) fprintf(stderr,
".")

#include "config.h"
#include <stdio.h>      /* */
#include <limits.h>
#include <math.h>
#include <ctype.h>
#include <signal.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#ifdef HP
#include <strings.h>
#endif
#if (defined(_POSIX_)||!defined(WIN32))   /* Change for Windows NT */
#include <unistd.h>
#include <sys/wait.h>
#include <sys/times.h>
#ifndef CLK_TCK
```

```
#define CLK_TCK _sysconf(3)     /* 3B2 clock ticks per second */
/* 3 is _SC_CLK_TCK */
#endif
#endif
#if (defined(WIN32)&&!defined(_POSIX_))
#include <process.h>
#include <signal.h>
#include <errno.h>

#endif

/* Ok, here are a few lines added for Teradata */
/* Some extra header files */
#include <stdlib.h>
#include <time.h>

#ifdef WIN32
/* Stuff needed to get process-specific times        */
typedef unsigned long DWORD;
typedef int BOOL;
#define WINAPI __stdcall
#define VOID void
#define DECLSPEC_IMPORT __declspec(dllimport)
#define STRICT
typedef __int64 LONGLONG;
typedef unsigned __int64 DWORDLONG;

typedef void *HANDLE;

#define WINBASEAPI DECLSPEC_IMPORT

typedef struct _FILETIME {
    DWORD dwLowDateTime;
    DWORD dwHighDateTime;
} FILETIME, *PFILETIME, *LPFILETIME;

WINBASEAPI
HANDLE
WINAPI
GetCurrentProcess(
    VOID
    );
WINBASEAPI
BOOL
WINAPI
GetProcessTimes(
    HANDLE hProcess,
    LPFILETIME lpCreationTime,
    LPFILETIME lpExitTime,
    LPFILETIME lpKernelTime,
    LPFILETIME lpUserTime
    );

#endif
char date[40];


#include <coptypes.h>  /* a Teradata specific include file */
/* undef VERSION which coptypes.h defined, since config.h will define
it*/
#undef VERSION

#define  EM_OK    0
#define  FILERR                400
#define  FILEOF                401
#define ROWSIZE 32750
#define MAXRECLEN 260

#ifdef WIN32
#pragma pack(1)
#endif

Int32 result;
typedef struct inmod_struct {
#ifdef PFAST
   Int16 Length;
#else
   Int32 ReturnCode;
   Int32 Length;
#endif
   char  Body[ROWSIZE];
} inmdtyp,*inmdptr;
inmdptr inmodptr;

#ifdef WIN32
#pragma pack(4)
#endif

#ifdef PFAST
char   DummyBuf[32767];



//FILE   *jobptr;
#define  EOBUF   -1
#endif
char   checkpointfile[255];


/* p1 is the pointer to the place in the parcel we are building
   where the next column should be copied. */
char *p1;

long i = 1;
long reccnt = 0;                              /* Record generated by
inmod */
long lineitem_in_order = 99;  /* State tracking */
long supplier_in_part = 99;
long nextrec = 0;                             /* next record to generate
*/

size_t templen;

long chkpnt;
FILE * f2;    /* For reading and writing checkpoint information */
```

```
#ifndef WIN32
struct tms starttime, endtime;   /* For timings */
time_t timeStart, timeEnd, usrtime, systime;
#else
FILETIME ProcCreated, ProcExited,     StartKernelTime, StartUserTime,
EndKernelTime, EndUserTime;
time_t timeStart, timeEnd;
double usrtime, systime;
#endif

/* End of Teradata specific changes, back to standard DBGEN driver.c
code */

#include "dss.h"
#include "dsstypes.h"
#include "bcd2.h"

/*
 * Function prototypes
 */
void usage(void);
int prep_direct(char *);
int close_direct(void);
void kill_load(void);
int pload(int tbl);
void gen_tbl(int tnum, long start, long count, long upd_num);
int pr_drange(int tbl, long min, long cnt, long num);
int set_files(int t, int pload);
void seed_name(char *tgt, long s, long c, long p);
int partial(int, int);


extern int      optind,
                opterr;
extern char     *optarg;
long rowcnt = 0,
    minrow = 0,
    upd_num = 0;
double flt_scale;



/*
 * general table descriptions. See dss.h for details on structure
 * NOTE: tables with no scaling info are scaled according to
 * another table
 *
 *
 * the following is based on the tdef structure defined in dss.h as:
 * typedef struct
 * {
 * char     *name;          -- name of the table;
 *                             flat file output in <name>.tbl
 * long      base;          -- base scale rowcount of table;
 *                             0 if derived
 * int       (*header) ();  -- function to prep output
 * int       (*loader[2]) ();  -- functions to present output
 * long      (*gen_seed) ();  -- functions to seed the RNG
 * int       child;           -- non-zero if there is an associated
                                 detail table
 * }         tdef;
 *
 */

/*
 * flat file print functions; used with -F(lat) option
 */
int pr_cust(customer_t * c, int mode);
int pr_line(order_t * o, int mode);
int pr_order(order_t * o, int mode);
int pr_part(part_t * p, int mode);
int pr_psupp(part_t * p, int mode);
int pr_supp(supplier_t * s, int mode);
int pr_order_line(order_t * o, int mode);
int pr_part_psupp(part_t * p, int mode);
int pr_time(dss_time_t * t, int mode);
int pr_nation(code_t * c, int mode);
int pr_region(code_t * c, int mode);

/*
 * inline load functions; used with -D(irect) option
 */
int ld_cust(customer_t * c, int mode);
int ld_line(order_t * o, int mode);
int ld_order(order_t * o, int mode);
int ld_part(part_t * p, int mode);
int ld_psupp(part_t * p, int mode);
int ld_supp(supplier_t * s, int mode);
int ld_order_line(order_t * o, int mode);
int ld_part_psupp(part_t * p, int mode);
int ld_time(dss_time_t * t, int mode);
int ld_nation(code_t * c, int mode);
int ld_region(code_t * c, int mode);

/*
 * seed generation functions; used with '-O s' option
 */
long sd_cust(long skip_count);
long sd_line(long skip_count);
long sd_order(long skip_count);
long sd_part(long skip_count);
long sd_psupp(long skip_count);
long sd_supp(long skip_count);
long sd_order_line(long skip_count);
long sd_part_psupp(long skip_count);
long sd_nation(long skip_count);
long sd_region(long skip_count);

/*
 * header output functions); used with -h(eader) option
 */
int hd_cust(FILE * f);
int hd_line(FILE * f);
int hd_order(FILE * f);
int hd_part(FILE * f);
int hd_psupp(FILE * f);
```

```
int hd_supp(FILE * f);
int hd_order_line(FILE * f);
int hd_part_psupp(FILE * f);
int hd_time(FILE * f);
int hd_nation(FILE * f);
int hd_region(FILE * f);

tdef            tdefs[] =
{
    {"part.tbl", "part table", 200000, hd_part,
     {pr_part, ld_part}, sd_part, NONE},
    {"partsupp.tbl", "partsupplier table", 200000, hd_psupp,
     {pr_psupp, ld_psupp}, sd_psupp, NONE},
    {"supplier.tbl", "suppliers table", 10000, hd_supp,
     {pr_supp, ld_supp}, sd_supp, NONE},
    {"customer.tbl", "customers table", 150000, hd_cust,
     {pr_cust, ld_cust}, sd_cust, NONE},
    {"order.tbl", "order table", 150000, hd_order,
     {pr_order, ld_order}, sd_order, NONE},
    {"lineitem.tbl", "lineitem table", 150000, hd_line,
     {pr_line, ld_line}, sd_line, NONE},
    {"order.tbl", "order/lineitem tables", 150000, hd_order_line,
     {pr_order_line, ld_order_line}, sd_order_line, LINE},
    {"part.tbl", "part/partsupplier tables", 200000, hd_part_psupp,
     {pr_part_psupp, ld_part_psupp}, sd_part_psupp, PSUPP},
    {"time.tbl", "time table", 2557, hd_time,
     {pr_time, ld_time}, NO_LFUNC, NONE},
    {"nation.tbl", "nation table", NATIONS_MAX, hd_nation,
     {pr_nation, ld_nation}, sd_nation, NONE},
    {"region.tbl", "region table", NATIONS_MAX, hd_region,
     {pr_region, ld_region}, sd_region, NONE},
    {"TPCDSEED", NULL, 0, NO_FUNC,
     {NO_FUNC, NO_FUNC}, NO_LFUNC, NONE}
};


int pids[MAX_CHILDREN];

void
mk_sparse(long *low_res, long *high_res, long base, long seq)
{
    long low_mask, seq_mask, overflow = 0;
    int count = 0;

    low_mask = (1<<SPARSE_KEEP) - 1;
    seq_mask = (1<<SPARSE_BITS) - 1;
    bin_bcd2(base, low_res, high_res);
    bcd2_div(low_res, high_res, 1 << SPARSE_KEEP);
    bcd2_mul(low_res, high_res, 1 << SPARSE_BITS);
    bcd2_add(low_res, high_res, seq);
    bcd2_mul(low_res, high_res, 1 << SPARSE_KEEP);
    bcd2_add(low_res, high_res, base & low_mask);
    bcd2_bin(&low_mask, *low_res);
    bcd2_bin(&seq_mask, *high_res);
    *low_res = low_mask;
    *high_res = seq_mask;
    return;
}
```

```
/*
 * routines to handle the graceful cleanup of multi-process loads
 */
/***** These routines from DBGEN driver.c are not used in Teradata
INMODs,
       so I've commented them out.
void
stop_proc(int signum)
{
exit(0);
}

void
kill_load(void)
{
    int i;

#if !defined(U2200) && !defined(DOS)
    for (i=0; i< MAX_CHILDREN; i++)
        if (pids[i])
            kill(SIGUSR1, pids[i]);
#endif /* !U2200 && !DOS
    return;
}   */

/*
 * re-set default output file names
 */
int
set_files(int i, int pload)
{
    char line[80],
         *new_name;

    if (table & (1 << i))
child_table:
        {
        if (pload != -1)
            sprintf(line, "%s.%d", tdefs[i].name, pload + 1);
        else
            {
            fprintf(stderr,"Enter new destination for %s data: ",
                tdefs[i].name);
            if (fgets(line, sizeof(line), stdin) == NULL)
                return(-1);;
            if ((new_name = strchr(line, '\n')) != NULL)
                *new_name = '\0';
            if (strlen(line) == 0)
                return(0);
            }
        new_name = (char *)malloc(strlen(line) + 1);
        MALLOC_CHECK(new_name);
        strcpy(new_name, line);
        tdefs[i].name = new_name;
        if (tdefs[i].child != NONE)
            {
```

```
            i = tdefs[i].child;
            tdefs[i].child = NONE;
            goto child_table;
            }
        }

    return(0);
}



/*
 * read the distributions needed in the benchamrk
 */
void
load_dists(void)
{
    read_dist(env_config(DIST_TAG, DIST_DFLT), "p_cntr", &p_cntr_set);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "colors", &colors);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "p_types",
&p_types_set);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "nations", &nations);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "regions", &regions);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "o_oprio",
                                            &o_priority_set);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "instruct",
                                           &l_instruct_set);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "smode", &l_smode_set);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "category",
                                           &l_category_set);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "rflag", &l_rflag_set);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "msegmnt", &c_mseg_set);

}



/*
 * generate a particular table
 */

/* This has been changed so that instead of generating the whole table,
   this routine generates one row for the table and then returns.
   nextrec is used to track which record we are generating.
   order and part must be static, since each contain multiple rows */
order_t         o;
part_t          part;
supplier_t      supp;
customer_t      cust;

void
gen_tbl(int tnum, long start, long count, long upd_num)
{
    long            i;



    dss_time_t      t;
    code_t code;
    long sk_low, sk_high;
    static long max_easy = LONG_MAX >> SPARSE_BITS;
    static int completed = 0;

        if (tnum == LINE)
                {
                        lineitem_in_order++;
                        if (o.lines>lineitem_in_order)
                                {
                                /* We already have it generated */
                                ld_line (&o, upd_num);
                                return;
                                }
                        else
                                lineitem_in_order=0;
                }
        else if (tnum == PSUPP)
                {
                        supplier_in_part++;
                        if (SUPP_PER_PART>supplier_in_part)
                                {
                                /* We already have it generated */
                                ld_psupp (&part, upd_num);
                                return;
                                }
                        else
                                supplier_in_part=0;
                }

        nextrec++;
    i = nextrec + start - 1;
    if (nextrec <= count)
        {
        LIFENOISE(10000);
        switch(tnum)
            {
            case LINE:
            case ORDER:
            case ORDER_LINE:
                if (i > max_easy)
                    mk_sparse(&sk_low, &sk_high, i,
                        (upd_num == 0)?0:1 + upd_num/(10000 /
refresh));
                else
                    {
                    sk_low = MK_SPARSE(i,
                        (upd_num == 0)?0:1 + upd_num/(10000 /
refresh));
                    sk_high = 0;
                    }
                mk_order(sk_low, sk_high, &o);
                tdefs[tnum].loader[direct] (&o, upd_num);
                break;
            case SUPP:
                mk_supp(i, &supp);
                tdefs[tnum].loader[direct] (&supp, upd_num);
                break;
```

```
            case CUST:
                mk_cust(i, &cust);
                tdefs[tnum].loader[direct] (&cust, upd_num);
                break;
            case PSUPP:
            case PART:
            case PART_PSUPP:
                mk_part(i, &part);
                tdefs[tnum].loader[direct] (&part, upd_num);
                break;
            case TIME:
                mk_time(i, &t);
                tdefs[tnum].loader[direct] (&t, 0);
                break;
            case NATION:
                mk_nation(i, &code);
                tdefs[tnum].loader[direct] (&code, 0);
                break;
            case REGION:
                mk_region(i, &code);
                tdefs[tnum].loader[direct] (&code, 0);
                break;
            }
        }
    /*completed |= 1 << tnum;  */
}



void
usage(void)
{
    /* This isn't a main(), so no usage stuff */
}

/*
 * pload() -- handle the parallel loading of tables
 */
#ifndef DOS

int
partial(int tbl, int s)
{

    char fname[80];

    rowcnt = tdefs[tbl].base * scale;
    if (rowcnt % children)
        {
        fprintf(stderr, "'-C' cannot split load equally\n");
        exit(-1);
        }
    else
        rowcnt /= children;
    /*if (verbose) */
        {
        fprintf(stderr, "Starting to load stage %d of %d of %s...",
            s + 1, children, tdefs[tbl].comment);
        }
    if (load_state(scale, children, s))
            {
            seed_name(fname, scale, children, s);
            fprintf(stderr, "Unable to load seeds (%s)\n", fname);
            exit(-1);
            }
    minrow = rowcnt * s + 1;
    if (direct == 0)
        set_files(tbl, s);

    return(0);
}

#ifdef NEVER
int
pload(int tbl)
{
        rowcnt = tdefs[tbl].base * scale;
    if (rowcnt % children)
        {
        fprintf(stderr, "'-C' cannot split load equally\n");
        exit(-1);
        }
    else
        rowcnt /= children;
    /* not used by the INMOD */
    return(0);
}
#endif /* !DOS */
#endif

int stream;
/* Because we aren't a main() program, we cant use getopt or options
   on the command line... So, we replace process_options with the
   ability to get some of this information from files on disk */
int
process_options(void)
{
    FILE * fptr;
    char    sclfile[] = "inmod.info";

    if ( !(fptr=fopen(sclfile, "r")) ) {
        fprintf(stderr, "open failed -- %s\n", sclfile);
               fprintf(stderr, "You MUST create this file, and specify
the scale factor\n");
               exit(1);
        //return(FILERR);
    }
       children = 1;
       step = -1;
    fscanf(fptr, "%lf %d %d\n", &flt_scale, &children, &step);
    fclose(fptr);

          if (flt_scale < 1.0)
           {
           int i;
```

```
            scale = 1;
            for (i=PART; i < TIME; i++)
                {
                tdefs[i].base = (long)(tdefs[i].base * flt_scale);
                if (tdefs[i].base < 1)
                    tdefs[i].base = 1;
                }
            }
        else
            scale = (long)flt_scale;
        if (scale < 1)
            {
            fprintf(stderr,
            "WARNING: Scale set to its lower bound (1)\n");
            scale = 1;
            }



    fprintf(stderr,"\n     SCALE = %f\n", flt_scale);

    if (children <= 1)
                children = 1;

    fprintf(stderr,"    children (total streams) = %d\n", children);
  if (children == 1)
        step = -1;

  fprintf(stderr,"      Step = %d\n", step);
  if (step >= 0)
                      step--;


  return(0);
}


/* The next routine is an extract of some, but not all, of the
   code from DBGEN's driver.c main() routine.  Because we are going
   to be a subroutine, we aren't going to have a main(), but we
   still need to run some of this initialization code */


int
main_stuff()
{
    int i;

 /*   table = (1 << CUST) |
            (1 << SUPP) |
            (1 << NATION) |
            (1 << REGION) |
            (1 << PART_PSUPP) |
            (1 << ORDER_LINE);               */
       /* This tells us what table to generate.  Inmods can only do
one at a time.*/
       table = (1 << TABLE);
    force = 0;
    verbose = 1;
    columnar = 0;
    bld_seeds = 0;
    header = 0;
    oldtime = 0;
    direct = 1;
    scale = 1;
    updates = 0;
    refresh = UPD_PCT;
    step = -1;

       /* Because we might need to reinitialize, let's set these here
*/
       tdefs[PART].base = 200000;
       tdefs[PSUPP].base = 200000;
       tdefs[SUPP].base = 10000;
       tdefs[CUST].base = 150000;
       tdefs[ORDER].base = 150000;
       tdefs[LINE].base = 150000;
       tdefs[ORDER_LINE].base = 150000;
       tdefs[PART_PSUPP].base = 200000;



    tdefs[ORDER].base *=
        ORDERS_PER_CUST; /* have to do this after init */
    tdefs[LINE].base *=
        ORDERS_PER_CUST; /* have to do this after init */
    tdefs[ORDER_LINE].base *=
        ORDERS_PER_CUST; /* have to do this after init */
    fnames = 0;
    db_name = NULL;
    gen_sql = 0;
    gen_rng = 0;
    children = 1;
       minrow = 1;

    i= process_options();
    if (i!=0)
               {
               fprintf(stderr, "Warning, initialization failed to work
right %d\n", i);
       /*return(i);*/
               }

    fprintf(stderr,
        "TPC-D Population Generator (Version %d.%d.%d%c)\n",
            DBGEN_RELEASE,DBGEN_VERSION,
DBGEN_MODIFICATION,DBGEN_PATCH);
    fprintf(stderr, "Copyright %s %s\n", TPC, C_DATES);


    load_dists();
    /* have to do this after init */
    tdefs[NATION].base = nations.count;
    tdefs[REGION].base = regions.count;
       o.lines = 0;
```

```
/**
 ** actual data generation section starts here
 **/
/*
 * open database connection or set all the file names, as appropriate
 */
    if (direct)
        prep_direct((db_name)?db_name:DBNAME);
    else
        if (fnames)
            for (i=PART; i <= REGION; i++)
                {
                if (table & (1 << i))
                    if (set_files(i, -1))
                        {
                        fprintf(stderr, "Load aborted!\n");
                        exit(1);
                        }
                }
/* So, now we are initialized, and ready to being the
   actual loading... At this point, we exit, since we
   are a subroutine which will get called once per row
   that needs to get generated */

    return (0);
}



/* Ok, that's it for source code taken from DBGEN's driver.c.
   After this point in the file, everything is Teradata specific
   code that is basic INMOD structure.  We don't do anything
   involving row generation after this point.  */



/*************************************************************
*
*   MakeRecord - Generate a record
*
*      Once a row has been successfully built and saved to a row
*      buffer, the row buffer is then copied to the main buffer
*      and returnd to Fastload.
*
*************************************************************/
int
MakeRecord()
{

   p1 = inmodptr->Body;
   reccnt++;   /* This counts which row we are generating */

   gen_tbl(TABLE, minrow, rowcnt, upd_num);

   /* end */
```

```
#ifdef PFAST
       inmodptr->Length = (Int16)(p1-(char *)inmodptr);
#else
       inmodptr->Length = p1-inmodptr->Body;
#endif

       if ((reccnt % 100000 == 0 )&&(inmodptr->Length>0))
      fprintf(stderr, " *** INMOD sent %d to fastload,
nextrec=%d\n",reccnt,nextrec);

#ifndef PFAST
       inmodptr->ReturnCode = 0;
       if (inmodptr->Length==0)
#else
       if (inmodptr->Length<=2)
#endif
      {
         if (inmodptr->Length<0 || inmodptr->Length > 32760)
              fprintf(stderr," !!! Bug!, inmodptr->Length =
%d\n",inmodptr->Length);

#ifdef PFAST
         if (inmodptr->Length >= MAXRECLEN)
              fprintf(stderr," !!! Bug!, inmodptr->Length =
%d\n",inmodptr->Length);
#endif
         inmodptr->Length = 0;
      reccnt--;
      fprintf(stderr," *** INMOD reached End of file\n");
      fprintf(stderr, " *** INMOD sent %d records to
fastload\n",reccnt);
         timeEnd = time(NULL);
      strftime(date, 35, "%a %b %d %H:%M:%S %Z %Y",
localtime(&timeEnd));
      fprintf(stderr," Finish making records: %s\n", date);


#ifndef WIN32
         if (times(&endtime) == -1) {
            fprintf(stderr,"times() fails\n");
#else
           if
(!GetProcessTimes(GetCurrentProcess(),&ProcCreated,&ProcExited,&EndKern
elTime,&EndUserTime)) {
              fprintf(stderr,"GetProcessTimes() fails\n");

#endif
            //exit(1);
         } else {
#ifndef WIN32
             usrtime = endtime.tms_utime - starttime.tms_utime;
             systime = endtime.tms_stime - starttime.tms_stime;
                       fprintf(stderr,"user time = %10.2lf secs,
system time = %10.2lf secs\n",
                  ((double) usrtime / CLK_TCK), ((double) systime /
CLK_TCK));
             fprintf(stderr,"time per request = %10.10lf secs \n",
```

```
                        ( ( (double) (usrtime + systime) / reccnt) /
CLK_TCK ));
#else
                        /* Times are in 100 nanosecond units! */
                      usrtime = (*(LONGLONG*)&EndUserTime -
*(LONGLONG*)&StartUserTime)/10000000.0;
                        systime = (*(LONGLONG*)&EndKernelTime -
*(LONGLONG*)&StartKernelTime)/10000000.0;
                        fprintf(stderr,"user time = %10.2f secs,
system time = %10.2lf secs\n",
                      ( usrtime ), ( systime ));
               fprintf(stderr,"time per request = %10.6lf microsecs \n",
                        ( ( (usrtime + systime)*1000000.0 / reccnt)
));
#endif

       }
         fprintf(stderr,"Clock time per request = %10.6lf millisecs
\n",difftime(timeEnd,timeStart)*1000.0/ reccnt);

#ifndef PFAST
      inmodptr->ReturnCode = FILEOF;
#endif
      return(FILEOF);
      }

   return(EM_OK);
}


Int32 OpenSource()
{
       int rc;

#ifdef PFAST
       if(getenv("HOSTNUMB") == NULL) {
              fprintf(stderr, "can not file HOSTNUMB var");
              return(FILEOF);
              }
        sprintf(checkpointfile, "./checkpoint.%ld",
atoi(getenv("HOSTNUMB")) );
        f2 = fopen(checkpointfile,"wb");
#else
    f2 = fopen("./checkpoint.dat","wb");
#endif

    if (f2==0)
        {
            fprintf(stderr,"Can't open the checkpoint file\n");
#ifndef PFAST
            inmodptr->ReturnCode = FILEOF;
#endif
            return(FILEOF);
        }

    reccnt = 0L;
        lineitem_in_order = 99;       /* State tracking */
        supplier_in_part = 99;
        nextrec = 0;

    fwrite(&reccnt,4,1,f2); /* mark that we are at the start */
    fclose(f2);
        f2 = NULL;

    rc = main_stuff();    /* Initialize everything */
        if (rc!=0)
               fprintf(stderr,"Main routine initialization failure\n");

        if (children > 1 && TABLE < TIME)
        {
               fprintf(stderr," Generating partial data for partitioned
load\n");
         if (step >= 0)
               partial(TABLE, step);
               else
                      {
                             fprintf(stderr," Don't know which step
I'm supposed to load\n");
                             exit(1);
                      }
        }
        else
        {
        if (TABLE < TIME)
          rowcnt = tdefs[TABLE].base * scale;
    else
          rowcnt = tdefs[TABLE].base;

        }
    if (verbose)
          fprintf(stderr, "Generating data for %s [pid: %d] minrow=%ld,
rowcnt=%ld\n",
                    tdefs[TABLE].comment, DSS_PROC, minrow, rowcnt);


               timeEnd = timeStart = time(NULL);
               strftime(date, 35, "%a %b %d %H:%M:%S %Z %Y",
localtime(&timeEnd));
#ifndef WIN32
         if (times(&starttime) == -1) {
                fprintf(stderr,"times() fails\n");
               // exit(1);
         }
#else
            if
(!GetProcessTimes(GetCurrentProcess(),&ProcCreated,&ProcExited,&StartKe
rnelTime,&StartUserTime)) {
                fprintf(stderr,"GetProcessTimes() fails\n");
                //exit(1);
         }
#endif


         fprintf(stderr,"\n  Start making records: %s\n", date);

        return(0);
```

```
}


/*************************************************************
 * HostRestart - Host restarted, rest and start sending data
 *               from the begining.
 *************************************************************/
Int32 HostRestart()
{
    int oldchkpnt;

#ifdef PFAST

          /* Notice how I use DummyBuf!  For restart calls the user
doesn't  */
          /* pass a big buffer for me to fill, I gotta use one on the
stack. */
          /* If I don't then GetReco() is gonna die like a stuck pig!
*/

      inmodptr = (struct inmod_struct *) DummyBuf;

        if(getenv("HOSTNUMB") == NULL) {
               fprintf(stderr, "can not file HOSTNUMB var");
               return(FILEOF);
               }
         sprintf(checkpointfile, "./checkpoint.%ld",
atoi(getenv("HOSTNUMB")) );
        f2 = fopen(checkpointfile,"rb");
#else
    f2 = fopen("./checkpoint.dat","rb");
#endif



                if (f2==0)
                   {
                       fprintf(stderr,"Can't open the checkpoint
file\n");
#ifndef PFAST
                       inmodptr->ReturnCode = FILEOF;
#endif
                       return(FILEOF);
                   }
                fread(&oldchkpnt,4,1,f2); /* mark that we are at the
start */
                fclose(f2);
                             f2 = NULL;

    chkpnt = oldchkpnt;

    result = OpenSource();
    if (result!=EM_OK)
        return(result);

    reccnt = 0;
       lineitem_in_order = 99;       /* State tracking */
       supplier_in_part = 99;
       nextrec = 0;

    while(reccnt < oldchkpnt)
        {
            result = MakeRecord();
            if (result!=EM_OK)
                return(result);
        }

    return(EM_OK);
}

/*************************************************************
 * CheckPoint - Save checkpoint
 *************************************************************/
Int32 CheckPoint()
{
    chkpnt = reccnt;

#ifdef PFAST
         f2 = fopen(checkpointfile,"wb");
#else
    f2 = fopen("./checkpoint.dat","wb");
#endif



                if (f2==0)
                   {
                       fprintf(stderr,"Can't open the checkpoint
file\n");
#ifndef PFAST
                       inmodptr->ReturnCode = FILEOF;
#endif
                       return(FILEOF);
                   }
                fwrite(&chkpnt,4,1,f2); /* mark that we are at the
start */
                fclose(f2);
                             f2 = NULL;
    return(EM_OK);
}

/*************************************************************
 * DBCRestart - DBC restarted, do what you have to do.
 *************************************************************/
Int32 DBCRestart()
{
int oldchkpnt;

#ifdef PFAST
          /* Notice how I use DummyBuf!  For restart calls the user
doesn't  */
          /* pass a big buffer for me to fill, I gotta use one on the
stack. */
```

```
         /* If I don't then GetReco() is gonna die like a stuck pig!
*/

     inmodptr = (struct inmod_struct *) DummyBuf;
        f2 = fopen(checkpointfile,"rb");
#else
    f2 = fopen("./checkpoint.dat","rb");
#endif



                if (f2==0)
                   {
                       fprintf(stderr,"Can't open the checkpoint
file\n");
#ifndef PFAST
                       inmodptr->ReturnCode = FILEOF;
#endif
                       return(FILEOF);
                   }
                fread(&oldchkpnt,4,1,f2); /* mark that we are at the
start */
                fclose(f2);
                             f2 = NULL;


    if (oldchkpnt!=chkpnt)
        {
            fprintf(stderr," **** Something is wrong trying to restart
at checkpoint\n");
        }

    result = OpenSource();
    if (result!=EM_OK)
        return(result);
    reccnt = 0;
    while(reccnt < oldchkpnt)
        {
            result = MakeRecord();
            if (result!=EM_OK)
                return(result);
        }


    return(EM_OK);
}

/*************************************************************
 * CleanUp - Do cleanup.
 *************************************************************/
Int32 CleanUp()
{
       if (verbose)
        fprintf(stderr, "done.\n");


    if (direct)
        close_direct();
        if (f2)
                fclose(f2);
    fprintf(stderr," *** INMOD Done\n");
    return(EM_OK);
}

/*************************************************************
 * InvalidCode - Invalid inmod code returned.
 *************************************************************/
Int32 InvalidCode()
{
    fprintf(stderr, "*** Invalid Inmod code\n");
    return(EM_OK);
}


/*************************************************************
*
*   BLKEXIT - Start processing
*
*      This is the main module which contains the checks for
*      number of records generated and buffer filling.  This
*      module also sends the filled buffer to the DBC.
*
*************************************************************/
#ifdef PFAST
Int32 BLKEXIT(int Function, char * tblptr, Int32 BufLen)
#else
Int32 BLKEXIT(char * tblptr)
#endif
{

   Int32 result;

   inmodptr = (struct inmod_struct *)tblptr;

#ifdef PFAST
        switch (Function) {
#else
    switch (inmodptr->ReturnCode) {
#endif

        case 0:  result=OpenSource();
                 if (EM_OK==result)
                    fprintf(stderr," *** INMOD starting....\n");
                 else
                    break;


        case 1:
#ifdef PFAST
                                 if (BufLen < MAXRECLEN + 2)
                                       return(EOBUF);             /* not
enough room in buffer */
#endif
                          result = MakeRecord();
                    break;
        case 2:
```

```
                        result = HostRestart();
                    break;
        case 3:  result = CheckPoint();
                    break;
        case 4:  result = DBCRestart();
                    break;
        case 5:  result = CleanUp();
                    break;
        default: result = InvalidCode();
    }

    return(result);
}
```

tdinload.c

```
/*************************************************************
 *  Title:      load_stub.c
 *  Sccsid:     @(#)load_stub.c       9.1.1.11     9/7/95  16:24:05
 *  Description:
 *              stub routines for:
 *          inline load of dss benchmark
 *          header creation for dss benchmark
 *
 *************************************************************
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#ifdef WIN32
#define INLINE _inline
#else
#define INLINE _Inline
#include <unistd.h>
#endif
#include "config.h"
#include "dss.h"
#include "dsstypes.h"

/* Start of Teradata specific code... These are the routines from
   load_stub.c, which are intended to be customized.  */

/*char buffer[32760]; */
/*extern*/
char * p1;

extern long reccnt;                             /* Record generated by
inmod */
extern long lineitem_in_order;        /* State tracking */
extern long supplier_in_part;
extern long nextrec;


/* First, some helper routines to copy data into the parcel we are
building */
INLINE
void DR_Strt()
{
/*      p1 = buffer + 2;*/     /* Skip 2 for the length field */
}
INLINE
void DR_End(int t)
{
        /*int len;

        reccnt++;

        *p1 = 0x0A; p1++;  /* Stick end-of-line in the buffer */
        /*
        len = (int)(p1-buffer);

        *((short *)buffer) =(short)(len-3); */ /* Fill in length */


}

INLINE
void DR_Int(long x)
{
        memcpy(p1,&x,4);
        p1 += 4;
}
INLINE
void DR_Int64(long xlow, long xhigh)    /* Used only for orderkey, which
gets > 32 bits at times */
{
        double tempx;
        unsigned long templ;
        long temph;
        if (xhigh==0)
                {
                        memcpy(p1,&xlow,4);
                        p1 += 4;
                        memcpy(p1,&xhigh,4);
                        p1 += 4;
                }
        else
                {
                        /* Sigh, don't have an Int64 type, so need to
use double and convert to 2 longs */
```

```
                        tempx = xhigh*10000000.0+xlow;
                        temph = (long)(tempx/4294967296.0);
                        templ = (unsigned long)(tempx-
temph*4294967296.0);
                        memcpy(p1,&templ,4);
                        p1 += 4;
                        memcpy(p1,&temph,4);
                        p1 += 4;

                }

}
INLINE
void DR_VStr(char * x)
{
        short len;
        len=(short)strlen(x);
        memcpy(p1,&len,2);
        p1 += 2;
        memcpy(p1,x,len);
        p1 += len;
}
INLINE
void DR_Str(char * x,size_t y)
{
        memset(p1,' ',y);
        memcpy(p1,x,strlen(x));
        p1+=y;
}
INLINE
void DR_Money(long x)
{
        long temp=0;

        if (x<0)
                temp = -1;
        /*sprintf(p1, "%13.2f", x * 0.01);
        p1+=13;  */
        memcpy(p1,&x,4);
        p1 += 4;
        memcpy(p1,&temp,4);
        p1 += 4;

}
INLINE
void DR_Char(char x)
{
        *p1 = x; p1++;
}
INLINE
void DR_Date(char * x)
{
        long tempdate;
        int y,m,d;
        /*memset(p1,' ',10);
        memcpy(p1,x,strlen(x));
        p1+=10;*/
        y=atoi(x);
        m=atoi(x+5);
        d=atoi(x+8);
/*      if (y<1800 || y>2200 || m<1 || m>12 || d<1 || d>31)
                {
                fprintf(stderr," Help! I can't interpret the date
%s\n",x);
                exit(1);
                } */
        tempdate=(y-1900)*10000+m*100+d;
        memcpy(p1,&tempdate,4);
        p1 += 4;

}

/*now the routines from load_stub.c */

int
close_direct(void)
{
        fprintf(stderr," **** Done with direct loading ****\n");
    return(0);
}

int
prep_direct(char * dbnamestr)
{

    /* any preload prep goes here */
    return(0);
}

int
hd_cust (FILE *f)
{
    static int count = 0;
        if (f);

    if (! count++)
        fprintf(stderr,"No header has been defined for the customer
table\n");

    return(0);
}

/*
CREATE TABLE TPCD.CUSTOMER ( C_CUSTKEY     INTEGER NOT NULL,
                             C_NAME        VARCHAR(25) NOT NULL,
                             C_ADDRESS     VARCHAR(40) NOT NULL,
                             C_NATIONKEY   INTEGER NOT NULL,
                             C_PHONE       CHAR(15) NOT NULL,
                             C_ACCTBAL     DECIMAL(15,2)   NOT NULL,
                             C_MKTSEGMENT  CHAR(10) NOT NULL,
                             C_COMMENT     VARCHAR(117) NOT NULL);
*/

int
ld_cust (customer_t *cp, int mode)
{
```

```
        if(mode);
        DR_Strt();
        DR_Int(cp->custkey);
    DR_VStr(cp->name);
        DR_VStr(cp->address);
        DR_Int(cp->nation_code);
        DR_Str(cp->phone,PHONE_LEN);
        DR_Money(cp->acctbal);
        DR_Str(cp->mktsegment,10);
        DR_VStr(cp->comment);
        DR_End(CUST);
        return(0);
}


int
hd_part (FILE *f)
{
    static int count = 0;
        if(f);

    if (! count++)
        fprintf(stderr,"No header has been defined for the part
table\n");

    return(0);
}

/*
CREATE TABLE TPCD.PART  ( P_PARTKEY     INTEGER NOT NULL,
                          P_NAME        VARCHAR(55) NOT NULL,
                          P_MFGR        CHAR(25) NOT NULL,
                          P_BRAND       CHAR(10) NOT NULL,
                          P_TYPE        VARCHAR(25) NOT NULL,
                          P_SIZE        INTEGER NOT NULL,
                          P_CONTAINER   CHAR(10) NOT NULL,
                          P_RETAILPRICE DECIMAL(15,2) NOT NULL,
                          P_COMMENT     VARCHAR(23) NOT NULL );
*/
int
ld_part (part_t *pp, int mode)
{

        if(mode);
        DR_Strt();
        DR_Int(pp->partkey);
        DR_VStr(pp->name);
        DR_Str(pp->mfgr,P_MFG_LEN);
        DR_Str(pp->brand,P_BRND_LEN);
        DR_VStr(pp->type);
        DR_Int(pp->size);
        DR_Str(pp->container,P_CNTR_LEN);
        DR_Money(pp->retailprice);
        DR_VStr(pp->comment);
        DR_End(PART);
        return(0);

}

/*
CREATE TABLE TPCD.PARTSUPP ( PS_PARTKEY     INTEGER NOT NULL,
                             PS_SUPPKEY     INTEGER NOT NULL,
                             PS_AVAILQTY    INTEGER NOT NULL,
                             PS_SUPPLYCOST  DECIMAL(15,2)  NOT NULL,
                             PS_COMMENT     VARCHAR(199) NOT NULL );
*/

int
ld_psupp (part_t *pp, int mode)
{
        int i;
        if(mode);
        i = supplier_in_part;
        /*for (i = 0; i < SUPP_PER_PART; i++) */
       {
                DR_Strt();
                DR_Int(pp->s[i].partkey);
                DR_Int(pp->s[i].suppkey);
                DR_Int(pp->s[i].qty);
                DR_Money(pp->s[i].scost);
                DR_VStr(pp->s[i].comment);
                DR_End(PSUPP);
       }
        return(0);

}

int
hd_time (FILE *f)
{
    static int count = 0;
        if(f);

    if (! count++)
        fprintf(stderr,"No header has been defined for the time
table\n");

    return(0);
}

/*
CREATE TABLE TPCD.TIME1    ( T_TIMEKEY     INTEGER NOT NULL,
--                           T_ALPHA       DATE NOT NULL,
--                           T_YEAR        INTEGER NOT NULL,
--                           T_MONTH       INTEGER NOT NULL,
--                           T_WEEK        INTEGER NOT NULL,
--                           T_DAY         INTEGER NOT NULL );
*/

int
ld_time (dss_time_t *tp, int mode)
{
    if(mode);

        DR_Strt();
        DR_Int(tp->timekey);
```

```
    DR_Date(tp->alpha);
    DR_Int(tp->year);
    DR_Int(tp->month);
    DR_Int(tp->week);
    DR_Int(tp->day);
       DR_End(TIME);

    return(0);
}


int
hd_supp (FILE *f)
{
    static int count = 0;
       if(f);

    if (! count++)
        fprintf(stderr,"No header has been defined for the supplier
table\n");

    return(0);
}

/*
CREATE TABLE TPCD.SUPPLIER ( S_SUPPKEY     INTEGER NOT NULL,
                             S_NAME        CHAR(25) NOT NULL,
                             S_ADDRESS     VARCHAR(40) NOT NULL,
                             S_NATIONKEY   INTEGER NOT NULL,
                             S_PHONE       CHAR(15) NOT NULL,
                             S_ACCTBAL     DECIMAL(15,2) NOT NULL,
                             S_COMMENT     VARCHAR(101) NOT NULL);
*/

int
ld_supp (supplier_t *sp, int mode)
{
       if(mode);
       DR_Strt();
       DR_Int(sp->suppkey);
    DR_Str(sp->name,25);
       DR_VStr(sp->address);
       DR_Int(sp->nation_code);
       DR_Str(sp->phone,15);
       DR_Money(sp->acctbal);
       DR_VStr(sp->comment);
       DR_End(SUPP);
    return(0);
}


int
hd_order (FILE *f)
{
    static int count = 0;
       if(f);

    if (! count++)
        fprintf(stderr,"No header has been defined for the order
table\n");

    return(0);
}

/*
CREATE TABLE TPCD.ORDER  ( O_ORDERKEY       INTEGER NOT NULL,
                           O_CUSTKEY        INTEGER NOT NULL,
                           O_ORDERSTATUS    CHAR(1) NOT NULL,
                           O_TOTALPRICE     DECIMAL(15,2) NOT NULL,
                           O_ORDERDATE      DATE NOT NULL,
                           O_ORDERPRIORITY  CHAR(15) NOT NULL,  -- R
                           O_CLERK          CHAR(15) NOT NULL,  -- R
                           O_SHIPPRIORITY   INTEGER NOT NULL,
                           O_COMMENT        VARCHAR(79) NOT NULL);
*/

int
ld_order (order_t *p, int mode)
{
       if(mode);
       DR_Strt();
       if (scale>300)
              {
                     DR_Int64(p->okey_low,p->okey_high);
              }
       else
              {
                     if (p->okey_high==0)
                     DR_Int(p->okey_low);
                     else
                            DR_Int(p->okey_high*10000000+p-
>okey_low);
              }
       DR_Int(p->custkey);
       DR_Char(p->orderstatus);
       DR_Money(p->totalprice);
       DR_Date(p->odate);
       DR_Str(p->opriority,O_OPRIO_LEN);
       DR_Str(p->clerk,O_CLRK_LEN);
       DR_Int(p->spriority);
       DR_VStr(p->comment);
       DR_End(ORDER);
    return(0);
}

/*
CREATE TABLE TPCD.LINEITEM ( L_ORDERKEY    INTEGER NOT NULL,
                             L_PARTKEY     INTEGER NOT NULL,
                             L_SUPPKEY     INTEGER NOT NULL,
                             L_LINENUMBER  INTEGER NOT NULL,
                             L_QUANTITY    DECIMAL(15,2) NOT NULL,
                             L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL,
                             L_DISCOUNT    DECIMAL(15,2) NOT NULL,
                             L_TAX         DECIMAL(15,2) NOT NULL,
                             L_RETURNFLAG  CHAR(1) NOT NULL,
                             L_LINESTATUS  CHAR(1) NOT NULL,
```

```
                                L_SHIPDATE    DATE NOT NULL,
                                L_COMMITDATE  DATE NOT NULL,
                                L_RECEIPTDATE DATE NOT NULL,
                                L_SHIPINSTRUCT CHAR(25) NOT NULL,  -- R
                                L_SHIPMODE     CHAR(10) NOT NULL,  -- R
                                L_COMMENT      VARCHAR(44) NOT NULL);
*/
ld_line (order_t *p, int mode)
{
        int i;
    if(mode);
        i = lineitem_in_order;
        /*for (i = 0; i < p->lines; i++)  */
         {
        DR_Strt();
        if (scale>300)
                {
                        DR_Int64(p->l[i].okey_low,p->l[i].okey_high);
                }
        else
                {
                        if (p->l[i].okey_high==0)
                        DR_Int(p->l[i].okey_low);
                        else
                                DR_Int(p->l[i].okey_high*10000000+p-
>l[i].okey_low);
                }
        DR_Int(p->l[i].partkey);
        DR_Int(p->l[i].suppkey);
        DR_Int(p->l[i].lcnt);
        DR_Int(p->l[i].quantity);
        DR_Money(p->l[i].eprice);
        DR_Money(p->l[i].discount);
        DR_Money(p->l[i].tax);
        DR_Char(p->l[i].rflag[0]);
        DR_Char(p->l[i].lstatus[0]);
        DR_Date(p->l[i].sdate);
        DR_Date(p->l[i].cdate);
        DR_Date(p->l[i].rdate);
        DR_Str(p->l[i].shipinstruct, L_INST_LEN);
        DR_Str(p->l[i].shipmode, L_SMODE_LEN);
        DR_VStr(p->l[i].comment);
        DR_End(LINE);
        }
    return(0);
}



int
hd_psupp (FILE *f)
{
    static int count = 0;
        if(f);

    if (! count++)
        fprintf(stderr,"%s %s\n",
            "No header has been defined for the",
                "part supplier table");

    return(0);
}


int
hd_line (FILE *f)
{
    static int count = 0;
        if(f);

    if (! count++)
        fprintf(stderr,"No header has been defined for the lineitem
table\n");

    return(0);
}

int
hd_nation (FILE *f)
{
    static int count = 0;
        if(f);

    if (! count++)
        fprintf(stderr,"No header has been defined for the nation
table\n");

    return(0);
}

/*
CREATE TABLE TPCD.NATION  ( N_NATIONKEY  INTEGER NOT NULL,
                            N_NAME       CHAR(25) NOT NULL,
                            N_REGIONKEY  INTEGER NOT NULL,
                            N_COMMENT    VARCHAR(152));
*/

int
ld_nation (code_t *cp, int mode)
{
        if(mode);
        DR_Strt();
        DR_Int(cp->code);
    DR_Str(cp->text,NATION_LEN);
        DR_Int(cp->join);
        DR_VStr(cp->comment);
        DR_End(NATION);
        return(0);
}

int
hd_region (FILE *f)
{
    static int count = 0;
        if(f);
```

```
    if (! count++)
        fprintf(stderr,"No header has been defined for the region
table\n");

    return(0);
}

/*
CREATE TABLE TPCD.REGION  ( R_REGIONKEY  INTEGER NOT NULL,
                            R_NAME       CHAR(25) NOT NULL,
                            R_COMMENT    VARCHAR(152));
*/

int
ld_region (code_t *cp, int mode)
{
        if(mode);
        DR_Strt();
        DR_Int(cp->code);
    DR_Str(cp->text,REGION_LEN);
        DR_Int(cp->join);
        DR_VStr(cp->comment);
        DR_End(REGION);

    return(0);
}

int
ld_order_line (order_t *p, int mode)
{

    ld_order(p, mode);
    ld_line (p, mode);

    return(0);
}

int
hd_order_line (FILE *f)
{

    hd_order(f);
    hd_line (f);

    return(0);
}

int
ld_part_psupp (part_t *p, int mode)
{
    ld_part(p, mode);
    ld_psupp (p, mode);

    return(0);
}

int
hd_part_psupp (FILE *f)
{
    hd_part(f);
    hd_psupp(f);

    return(0);
}
```

F.2 Source Code for Pipeline-to-FastLoad Approach

In this approach to loading, DBGEN is basically unmodified from the standard TPC distribution. All that has been done is to replace the load_stub.c file with one specific for Teradata.

Makefile

```
#  Sccsid:     @(#)makefile.suite     9.1.1.13     12/14/95  16:06:25
# makefile for DSS benchamrk data generator
#
#
#################
## CHANGE NAME OF ANSI COMPILER HERE
#################
CC      = cc
# Current values for DATABASE are: INFORMIX, DB2, TDAT (Teradata),
#                                  SQLSERVER, SYBASE
# Current values for PLATFORM are: ATT, DOS, HP, IBM, ICL, MVS,
#                                  SGI, U2200, VMS, WIN32
DATABASE=TDAT
PLATFORM=ATT
#
CFLAGS  = -O -D$(DATABASE) -D$(PLATFORM) -DSTDLIB_HAS_GETOPT -D__STDC__
-o1 -g -586 -I/usr/include -I/usr/ucbinclude

LDFLAGS =
# The OBJ,EXE and LIB macros will need to be changed for compilation
under
#  Windows NT
OBJ     = .o
EXE     =
LIBS    = -lcliv2 -ltdusr -lfast -lsocket -lnsl  -lm -lc -lgen -lx
#
# NO CHANGES SHOULD BE NECESSARY BELOW THIS LINE
################

PROG1 = dbgen$(EXE)
PROG2 = qgen$(EXE)
PROG3 = tpcddriver

HDR1 = dss.h rnd.h config.h dsstypes.h shared.h bcd2.h
SRC1 = build.c driver.c bm_utils.c rnd.c print.c load_stub.c bcd2.c
speed_seed.c
```

```
OBJ1 = build$(OBJ) driver$(OBJ) bm_utils$(OBJ) rnd$(OBJ) print$(OBJ)
load_stub$(OBJ) bcd2$(OBJ) speed_seed$(OBJ)
SETS = dists.dss
DDL  = dss.ddl dss.ri
OTHER=makefile.suite $(SETS) $(DDL) 10001001
DOC=README History Porting.Notes FAQ
DBGENSRC=$(SRC1) $(HDR1) $(OTHER) $(DOC) $(SRC2) $(HDR2) $(SRC3)
$(HDR3)
SRC2 = qgen.c varsub.c
HDR2 = tpcd.h
OBJ2 = build$(OBJ) bm_utils$(OBJ) qgen$(OBJ) rnd$(OBJ) varsub$(OBJ)
HDR3 = dss.h rnd.h config.h dsstypes.h shared.h bcd2.h tdatsql.h
tsqlt.h
SRC3 = build.c tpcddriver.c bm_utils.c rnd.c vsub.c bcd2.c tdatsql.c
tsqlt.c tdatload.c
OBJ3 = build$(OBJ) tpcddriver$(OBJ) bm_utils$(OBJ) rnd$(OBJ) vsub$(OBJ)
bcd2$(OBJ) tdatsql$(OBJ) tsqlt$(OBJ) tdatload$(OBJ)


all: $(PROG1) $(PROG2) $(PROG3)

$(PROG1): $(OBJ1) $(SETS)
        $(CC) $(LDFLAGS) -o $@ $(OBJ1) $(LIBS)

$(PROG2): $(OBJ2)
        $(CC) $(LDFLAGS) -o $@ $(OBJ2) $(LIBS)

$(PROG3): $(OBJ3) $(SETS)
        $(CC) $(LDFLAGS) -o $@ $(OBJ3) $(LIBS)

clean:
        rm -f  $(PROG1) $(PROG2) $(PROG3) $(OBJ1) $(OBJ2) $(OBJ3)
lint:
        lint $(CFLAGS) $(LDFLAGS) $(SRC3)
tar: $(DBGENSRC)
        tar cvhf $(PROG1).tar $(DBGENSRC)
dbgenshar: $(DBGENSRC)
        shar -o dbgen.shar $(DBGENSRC)
dbgendisk: $(DBGENSRC)
        for f in $(DBGENSRC) ; \
        do  \
        unix2dos $$f /pcfs/$$f ; \
        done
portable:
        for f in $(DBGENSRC) ; \
        do  \
        expand $$f > /tmp/foo; \
        mv -f /tmp/foo $$f; \
        awk 'length > 72 { print FILENAME ":" NR " too long " }' $$f ;
\
        done
rnd$(OBJ): rnd.h
$(OBJ1): config.h makefile.comp
$(OBJ1): dss.h dsstypes.h
$(OBJ2): dss.h tpcd.h config.h
$(OBJ3): dss.h tpcd.h config.h dsstypes.h tdatsql.h
```

tdatload.c

```
/*************************************************************
 *  Title:      load_stub.c
 *  Sccsid:     @(#)load_stub.c      9.1.1.11     9/7/95  16:24:05
 *  Description:
 *              stub routines for:
 *          inline load of dss benchmark
 *          header creation for dss benchmark
 *
 *************************************************************
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#ifdef WIN32
#include <process.h>
#define WIN32_LEAN_AND_MEAN
#define NOATOM
#define NOGDICAPMASKS
#define NOMETAFILE
#define NOMINMAX
#define NOMSG
#define NOOPENFILE
#define NORASTEROPS
#define NOSCROLL
#define NOSOUND
#define NOSYSMETRICS
#define NOTEXTMETRIC
#define NOWH
#define NOCOMM
#define NOKANJI
#define NOMCX
#define INLINE _inline
#pragma warning(disable:4201)
#pragma warning(disable:4214)
#pragma warning(disable:4514)
#include <windows.h>
#else
#define INLINE _Inline
#include <unistd.h>
#include <sys/wait.h>
#endif
#include "config.h"
#include "dss.h"
#include "dsstypes.h"

/* Start of Teradata specific code... These are the routines from
   load_stub.c, which are intended to be customized.  */
int prep_direct(char *);


#ifndef WIN32
```

```
FILE * pipefile[20];
#else
HANDLE pipehandle[20];
HANDLE processhandle[20];
#endif

char logonstr[256] = { 0 };
char pipename[120];
char fastname[120];
char buffer[32760];
/*extern*/
char * p1;
int reccnt = 0;
int loadinprogress = 0;
int fastloadsstarted = 0;


void WriteFastloadJob(int t,int mode)
{

  FILE * fastjob;
  char tablename[60];

  tablename[0] ='\0';
  if (t==19)
  {
         strcpy(tablename,"KEYS_TO_DELETE");
  }
  else
  switch(t)    {
    case PART: strcpy(tablename,"PARTTBL"); break;
    case PSUPP:  strcpy(tablename,"PARTSUPP");  break;
    case  SUPP: strcpy(tablename,"SUPPLIER"); break;
        case  CUST: strcpy(tablename,"CUSTOMER"); break;
        case  ORDER: strcpy(tablename,"ORDERTBL");  break;
        case  LINE: strcpy(tablename,"LINEITEM");  break;
        case  NATION: strcpy(tablename,"NATION");  break;
        case  REGION: strcpy(tablename,"REGION");  break;
    case  TIME: strcpy(tablename,"TIMETBL"); break;
        case  ORDER_LINE:
        case  PART_PSUPP:
        case  NONE:
        default: fprintf(stderr," Illegal table to load\n");
                exit(1);
        }
if (mode>0 && t!=19)
        strcat(tablename,"_UPDATES");

if (logonstr[0] == '\0' && db_name[0] != '\0')
        prep_direct(db_name);

if (verbose)
        fprintf(stderr," Opening %s for write\n",fastname);
fastjob = fopen(fastname,"w");
if (fastjob==NULL)
        {
                fprintf(stderr," Cannot create the fastload input job
stream for %s\n",fastname);
                return;
        }
if (fprintf(fastjob,"\n")<=0)
        fprintf(stderr," Cannot write the fastload input job
stream\n");
fprintf(fastjob,"\n");
fprintf(fastjob,"SESSIONS 30;\n");
fprintf(fastjob,"LOGON %s;\n",logonstr);
fprintf(fastjob,"\n");
fprintf(fastjob,"DROP TABLE %s;\n",tablename);
fprintf(fastjob,"DROP TABLE ERR%s1;\n",tablename);
fprintf(fastjob,"DROP TABLE ERR%s2;\n",tablename);
fprintf(fastjob,"\n");
fprintf(fastjob,"CREATE MULTISET TABLE %s\n",tablename);
fprintf(fastjob,"  (\n");
switch(t) {
case 19:
if (scale>300)
fprintf(fastjob,"    ORDERKEY       DECIMAL(15,0) NOT NULL\n");
else
fprintf(fastjob,"    ORDERKEY       INTEGER NOT NULL\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"UNIQUE PRIMARY INDEX( ORDERKEY );\n");
break;
case CUST:
fprintf(fastjob,"    C_CUSTKEY     INTEGER NOT NULL\n");
fprintf(fastjob,"   ,C_NAME        VARCHAR(25) NOT NULL\n");
fprintf(fastjob,"   ,C_ADDRESS     VARCHAR(40) NOT NULL\n");
fprintf(fastjob,"   ,C_NATIONKEY   INTEGER NOT NULL\n");
fprintf(fastjob,"   ,C_PHONE       CHAR(15) NOT NULL\n");
fprintf(fastjob,"   ,C_ACCTBAL     DECIMAL(15,2) NOT NULL\n");
fprintf(fastjob,"   ,C_MKTSEGMENT  CHAR(10) NOT NULL\n");
fprintf(fastjob,"   ,C_COMMENT     VARCHAR(117) NOT NULL\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"UNIQUE PRIMARY INDEX( C_CUSTKEY );\n");
break;
case LINE:
if (scale>300)
fprintf(fastjob,"    L_ORDERKEY       DECIMAL(15,0) NOT NULL\n");
else
fprintf(fastjob,"    L_ORDERKEY       INTEGER NOT NULL\n");
fprintf(fastjob,"   ,L_PARTKEY        INTEGER NOT NULL\n");
fprintf(fastjob,"   ,L_SUPPKEY        INTEGER NOT NULL\n");
fprintf(fastjob,"   ,L_LINENUMBER     INTEGER NOT NULL\n");
fprintf(fastjob,"   ,L_QUANTITY       DECIMAL(15,2) NOT NULL\n");   /*
Was INTEGER */
fprintf(fastjob,"   ,L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL\n");
fprintf(fastjob,"   ,L_DISCOUNT       DECIMAL(15,2) NOT NULL\n");
fprintf(fastjob,"   ,L_TAX            DECIMAL(15,2) NOT NULL\n");
fprintf(fastjob,"   ,L_RETURNFLAG     CHAR(1) NOT NULL\n");
fprintf(fastjob,"   ,L_LINESTATUS     CHAR(1) NOT NULL\n");
fprintf(fastjob,"   ,L_SHIPDATE       DATE FORMAT 'yyyy-mm-dd' NOT
NULL\n");
fprintf(fastjob,"   ,L_COMMITDATE     DATE FORMAT 'yyyy-mm-dd' NOT
NULL\n");
fprintf(fastjob,"   ,L_RECEIPTDATE    DATE FORMAT 'yyyy-mm-dd' NOT
NULL\n");
fprintf(fastjob,"   ,L_SHIPINSTRUCT   CHAR(25) NOT NULL\n");
```

```
fprintf(fastjob,"   ,L_SHIPMODE       CHAR(10) NOT NULL\n");
fprintf(fastjob,"   ,L_COMMENT        VARCHAR(44) NOT NULL\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,";\n");
break;
case NATION:
fprintf(fastjob,"    N_NATIONKEY  INTEGER NOT NULL\n");
fprintf(fastjob,"   ,N_NAME       CHAR(25) NOT NULL\n");
fprintf(fastjob,"   ,N_REGIONKEY  INTEGER NOT NULL\n");
fprintf(fastjob,"   ,N_COMMENT    VARCHAR(152) NOT NULL\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"UNIQUE PRIMARY INDEX( N_NAME )\n");
fprintf(fastjob,";\n");
break;
case ORDER:
if (scale>300)
fprintf(fastjob,"    O_ORDERKEY       DECIMAL(15,0) NOT NULL\n");
else
fprintf(fastjob,"    O_ORDERKEY       INTEGER NOT NULL\n");
fprintf(fastjob,"   ,O_CUSTKEY        INTEGER NOT NULL\n");
fprintf(fastjob,"   ,O_ORDERSTATUS    CHAR(1) NOT NULL\n");
fprintf(fastjob,"   ,O_TOTALPRICE     DECIMAL(15,2) NOT NULL\n");
fprintf(fastjob,"   ,O_ORDERDATE      DATE FORMAT 'yyyy-mm-dd' NOT
NULL\n");
fprintf(fastjob,"   ,O_ORDERPRIORITY  VARCHAR(15) NOT NULL\n");
fprintf(fastjob,"   ,O_CLERK          VARCHAR(15) NOT NULL\n");
fprintf(fastjob,"   ,O_SHIPPRIORITY   INTEGER NOT NULL\n");
fprintf(fastjob,"   ,O_COMMENT        VARCHAR(79) NOT NULL\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"UNIQUE PRIMARY INDEX( O_ORDERKEY );\n");
break;
case PART:
fprintf(fastjob,"    P_PARTKEY      INTEGER NOT NULL\n");
fprintf(fastjob,"   ,P_NAME         VARCHAR(55) NOT NULL\n");
fprintf(fastjob,"   ,P_MFGR         CHAR(25) NOT NULL\n");
fprintf(fastjob,"   ,P_BRAND        CHAR(10) NOT NULL\n");
fprintf(fastjob,"   ,P_TYPE         VARCHAR(25) NOT NULL\n");
fprintf(fastjob,"   ,P_SIZE         INTEGER NOT NULL\n");
fprintf(fastjob,"   ,P_CONTAINER    CHAR(10) NOT NULL\n");
fprintf(fastjob,"   ,P_RETAILPRICE  DECIMAL(15,2) NOT NULL\n");
fprintf(fastjob,"   ,P_COMMENT      VARCHAR(23) NOT NULL\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"UNIQUE PRIMARY INDEX( P_PARTKEY );\n");
break;
case PSUPP:
fprintf(fastjob,"    PS_PARTKEY     INTEGER NOT NULL\n");
fprintf(fastjob,"   ,PS_SUPPKEY     INTEGER NOT NULL\n");
fprintf(fastjob,"   ,PS_AVAILQTY    INTEGER NOT NULL\n");
fprintf(fastjob,"   ,PS_SUPPLYCOST  DECIMAL(15,2) NOT NULL\n");
fprintf(fastjob,"   ,PS_COMMENT     VARCHAR(199) NOT NULL\n");
fprintf(fastjob,"  )\n");
/* Default to non-unique primary index on PS_PARTKEY */
fprintf(fastjob,";\n");
break;
case REGION:
fprintf(fastjob,"    R_REGIONKEY  INTEGER NOT NULL\n");
fprintf(fastjob,"   ,R_NAME       CHAR(25) NOT NULL\n");
fprintf(fastjob,"   ,R_COMMENT    VARCHAR(152) NOT NULL\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"UNIQUE PRIMARY INDEX( R_REGIONKEY );\n");
break;
case SUPP:
fprintf(fastjob,"    S_SUPPKEY    INTEGER NOT NULL\n");
fprintf(fastjob,"   ,S_NAME       CHAR(25) NOT NULL\n");
fprintf(fastjob,"   ,S_ADDRESS    VARCHAR(40) NOT NULL\n");
fprintf(fastjob,"   ,S_NATIONKEY  INTEGER NOT NULL\n");
fprintf(fastjob,"   ,S_PHONE      CHAR(15) NOT NULL\n");
fprintf(fastjob,"   ,S_ACCTBAL    DECIMAL(15,2) NOT NULL\n");
fprintf(fastjob,"   ,S_COMMENT    VARCHAR(101) NOT NULL\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"UNIQUE PRIMARY INDEX( S_SUPPKEY );\n");
break;
}

fprintf(fastjob,"\n");
fprintf(fastjob,"BEGIN LOADING %s\n",tablename);
fprintf(fastjob,"      ERRORFILES ERR%s1,
ERR%s2;\n",tablename,tablename);
fprintf(fastjob,"\n");
fprintf(fastjob,"DEFINE\n");
switch (t) {
case 19:
if (scale>300)
fprintf(fastjob,"    F_ORDERKEY       DECIMAL(15,0)\n");
else
fprintf(fastjob,"    F_ORDERKEY       (INTEGER)\n");
break;
case CUST:
fprintf(fastjob,"   F_C_CUSTKEY     (INTEGER)\n");
fprintf(fastjob,"  ,F_C_NAME        (VARCHAR(25))\n");
fprintf(fastjob,"  ,F_C_ADDRESS     (VARCHAR(40))\n");
fprintf(fastjob,"  ,F_C_NATIONKEY   (INTEGER)\n");
fprintf(fastjob,"  ,F_C_PHONE       (CHAR(15))\n");
fprintf(fastjob,"  ,F_C_ACCTBAL     (DECIMAL(15,2))\n");
fprintf(fastjob,"  ,F_C_MKTSEGMENT  (CHAR(10))\n");
fprintf(fastjob,"  ,F_C_COMMENT     (VARCHAR(117))\n");
break;
case LINE:
if (scale>300)
fprintf(fastjob,"    F_L_ORDERKEY       DECIMAL(15,0)\n");
else
fprintf(fastjob,"    F_L_ORDERKEY       (INTEGER)\n");
fprintf(fastjob,"   ,F_L_PARTKEY        (INTEGER)\n");
fprintf(fastjob,"   ,F_L_SUPPKEY        (INTEGER)\n");
fprintf(fastjob,"   ,F_L_LINENUMBER     (INTEGER)\n");
fprintf(fastjob,"   ,F_L_QUANTITY       (INTEGER)\n");
fprintf(fastjob,"   ,F_L_EXTENDEDPRICE  (DECIMAL(15,2))\n");
fprintf(fastjob,"   ,F_L_DISCOUNT       (DECIMAL(15,2))\n");
fprintf(fastjob,"   ,F_L_TAX            (DECIMAL(15,2))\n");
fprintf(fastjob,"   ,F_L_RETURNFLAG     (CHAR(1))\n");
fprintf(fastjob,"   ,F_L_LINESTATUS     (CHAR(1))\n");
fprintf(fastjob,"   ,F_L_SHIPDATE       (DATE)\n");
fprintf(fastjob,"   ,F_L_COMMITDATE     (DATE)\n");
fprintf(fastjob,"   ,F_L_RECEIPTDATE    (DATE)\n");
fprintf(fastjob,"   ,F_L_SHIPINSTRUCT   (CHAR(25))\n");
fprintf(fastjob,"   ,F_L_SHIPMODE       (CHAR(10))\n");
```

```
fprintf(fastjob,"   ,F_L_COMMENT        (VARCHAR(44))\n");
break;
case NATION:
fprintf(fastjob,"   F_N_NATIONKEY  (INTEGER)\n");
fprintf(fastjob,"  ,F_N_NAME       (CHAR(25))\n");
fprintf(fastjob,"  ,F_N_REGIONKEY  (INTEGER)\n");
fprintf(fastjob,"  ,F_N_COMMENT    (VARCHAR(152))\n");
break;
case ORDER:
if (scale>300)
fprintf(fastjob,"    F_O_ORDERKEY       DECIMAL(15,0)\n");
else
fprintf(fastjob,"    F_O_ORDERKEY       (INTEGER)\n");
fprintf(fastjob,"   ,F_O_CUSTKEY        (INTEGER)\n");
fprintf(fastjob,"   ,F_O_ORDERSTATUS    (CHAR(1))\n");
fprintf(fastjob,"   ,F_O_TOTALPRICE     (DECIMAL(15,2))\n");
fprintf(fastjob,"   ,F_O_ORDERDATE      (DATE)\n");
fprintf(fastjob,"   ,F_O_ORDERPRIORITY  (CHAR(15))\n");
fprintf(fastjob,"   ,F_O_CLERK          (CHAR(15))\n");
fprintf(fastjob,"   ,F_O_SHIPPRIORITY   (INTEGER)\n");
fprintf(fastjob,"   ,F_O_COMMENT        (VARCHAR(79))\n");
break;
case PART:
fprintf(fastjob,"    F_P_PARTKEY      (INTEGER)\n");
fprintf(fastjob,"   ,F_P_NAME         (VARCHAR(55))\n");
fprintf(fastjob,"   ,F_P_MFGR         (CHAR(25))\n");
fprintf(fastjob,"   ,F_P_BRAND        (CHAR(10))\n");
fprintf(fastjob,"   ,F_P_TYPE         (VARCHAR(25))\n");
fprintf(fastjob,"   ,F_P_SIZE         (INTEGER)\n");
fprintf(fastjob,"   ,F_P_CONTAINER    (CHAR(10))\n");
fprintf(fastjob,"   ,F_P_RETAILPRICE  (DECIMAL(15,2))\n");
fprintf(fastjob,"   ,F_P_COMMENT      (VARCHAR(23))\n");
break;
case PSUPP:
fprintf(fastjob,"    F_PS_PARTKEY     (INTEGER)\n");
fprintf(fastjob,"   ,F_PS_SUPPKEY     (INTEGER)\n");
fprintf(fastjob,"   ,F_PS_AVAILQTY    (INTEGER)\n");
fprintf(fastjob,"   ,F_PS_SUPPLYCOST  (DECIMAL(15,2))\n");
fprintf(fastjob,"   ,F_PS_COMMENT     (VARCHAR(199))\n");
break;
case REGION:
fprintf(fastjob,"   F_R_REGIONKEY  (INTEGER)\n");
fprintf(fastjob,"  ,F_R_NAME       (CHAR(25))\n");
fprintf(fastjob,"  ,F_R_COMMENT    (VARCHAR(152))\n");
break;
case SUPP:
fprintf(fastjob,"   F_S_SUPPKEY    (INTEGER)\n");
fprintf(fastjob,"  ,F_S_NAME       (CHAR(25))\n");
fprintf(fastjob,"  ,F_S_ADDRESS    (VARCHAR(40))\n");
fprintf(fastjob,"  ,F_S_NATIONKEY  (INTEGER)\n");
fprintf(fastjob,"  ,F_S_PHONE      (CHAR(15))\n");
fprintf(fastjob,"  ,F_S_ACCTBAL    (DECIMAL(15,2))\n");
fprintf(fastjob,"  ,F_S_COMMENT    (VARCHAR(101))\n");
break;
}

fprintf(fastjob,"FILE=%s;\n",pipename);
fprintf(fastjob,"\n");
fprintf(fastjob,"SHOW;\n");
fprintf(fastjob,"\n");
fprintf(fastjob,"INSERT INTO %s\n",tablename);
fprintf(fastjob,"  (\n");
switch (t) {
case 19:
fprintf(fastjob,"    ORDERKEY\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"VALUES\n");
fprintf(fastjob,"  (\n");
fprintf(fastjob,"   :F_ORDERKEY\n");
break;
case CUST:
fprintf(fastjob,"    C_CUSTKEY\n");
fprintf(fastjob,"   ,C_NAME\n");
fprintf(fastjob,"   ,C_ADDRESS\n");
fprintf(fastjob,"   ,C_NATIONKEY\n");
fprintf(fastjob,"   ,C_PHONE\n");
fprintf(fastjob,"   ,C_ACCTBAL\n");
fprintf(fastjob,"   ,C_MKTSEGMENT\n");
fprintf(fastjob,"   ,C_COMMENT\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"VALUES\n");
fprintf(fastjob,"  (\n");
fprintf(fastjob,"   :F_C_CUSTKEY,\n");
fprintf(fastjob,"   :F_C_NAME,\n");
fprintf(fastjob,"   :F_C_ADDRESS,\n");
fprintf(fastjob,"   :F_C_NATIONKEY,\n");
fprintf(fastjob,"   :F_C_PHONE,\n");
fprintf(fastjob,"   :F_C_ACCTBAL,\n");
fprintf(fastjob,"   :F_C_MKTSEGMENT,\n");
fprintf(fastjob,"   :F_C_COMMENT\n");
break;
case LINE:
fprintf(fastjob,"    L_ORDERKEY\n");
fprintf(fastjob,"   ,L_PARTKEY\n");
fprintf(fastjob,"   ,L_SUPPKEY\n");
fprintf(fastjob,"   ,L_LINENUMBER\n");
fprintf(fastjob,"   ,L_QUANTITY\n");
fprintf(fastjob,"   ,L_EXTENDEDPRICE\n");
fprintf(fastjob,"   ,L_DISCOUNT\n");
fprintf(fastjob,"   ,L_TAX\n");
fprintf(fastjob,"   ,L_RETURNFLAG\n");
fprintf(fastjob,"   ,L_LINESTATUS\n");
fprintf(fastjob,"   ,L_SHIPDATE\n");
fprintf(fastjob,"   ,L_COMMITDATE\n");
fprintf(fastjob,"   ,L_RECEIPTDATE\n");
fprintf(fastjob,"   ,L_SHIPINSTRUCT\n");
fprintf(fastjob,"   ,L_SHIPMODE\n");
fprintf(fastjob,"   ,L_COMMENT\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"VALUES\n");
fprintf(fastjob,"  (\n");
fprintf(fastjob,"   :F_L_ORDERKEY,\n");
fprintf(fastjob,"   :F_L_PARTKEY,\n");
fprintf(fastjob,"   :F_L_SUPPKEY,\n");
fprintf(fastjob,"   :F_L_LINENUMBER,\n");
```

```
fprintf(fastjob,"   :F_L_QUANTITY,\n");
fprintf(fastjob,"   :F_L_EXTENDEDPRICE,\n");
fprintf(fastjob,"   :F_L_DISCOUNT,\n");
fprintf(fastjob,"   :F_L_TAX,\n");
fprintf(fastjob,"   :F_L_RETURNFLAG,\n");
fprintf(fastjob,"   :F_L_LINESTATUS,\n");
fprintf(fastjob,"   :F_L_SHIPDATE,\n");
fprintf(fastjob,"   :F_L_COMMITDATE,\n");
fprintf(fastjob,"   :F_L_RECEIPTDATE,\n");
fprintf(fastjob,"   :F_L_SHIPINSTRUCT,\n");
fprintf(fastjob,"   :F_L_SHIPMODE,\n");
fprintf(fastjob,"   :F_L_COMMENT\n");
break;
case NATION:
fprintf(fastjob,"    N_NATIONKEY\n");
fprintf(fastjob,"   ,N_NAME\n");
fprintf(fastjob,"   ,N_REGIONKEY\n");
fprintf(fastjob,"   ,N_COMMENT\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"VALUES\n");
fprintf(fastjob,"  (\n");
fprintf(fastjob,"   :F_N_NATIONKEY,\n");
fprintf(fastjob,"   :F_N_NAME,\n");
fprintf(fastjob,"   :F_N_REGIONKEY,\n");
fprintf(fastjob,"   :F_N_COMMENT\n");
break;
case ORDER:
fprintf(fastjob,"    O_ORDERKEY\n");
fprintf(fastjob,"   ,O_CUSTKEY\n");
fprintf(fastjob,"   ,O_ORDERSTATUS\n");
fprintf(fastjob,"   ,O_TOTALPRICE\n");
fprintf(fastjob,"   ,O_ORDERDATE\n");
fprintf(fastjob,"   ,O_ORDERPRIORITY\n");
fprintf(fastjob,"   ,O_CLERK\n");
fprintf(fastjob,"   ,O_SHIPPRIORITY\n");
fprintf(fastjob,"   ,O_COMMENT\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"VALUES\n");
fprintf(fastjob,"  (\n");
fprintf(fastjob,"   :F_O_ORDERKEY,\n");
fprintf(fastjob,"   :F_O_CUSTKEY,\n");
fprintf(fastjob,"   :F_O_ORDERSTATUS,\n");
fprintf(fastjob,"   :F_O_TOTALPRICE,\n");
fprintf(fastjob,"   :F_O_ORDERDATE,\n");
fprintf(fastjob,"   :F_O_ORDERPRIORITY,\n");
fprintf(fastjob,"   :F_O_CLERK,\n");
fprintf(fastjob,"   :F_O_SHIPPRIORITY,\n");
fprintf(fastjob,"   :F_O_COMMENT\n");
break;
case PART:
fprintf(fastjob,"    P_PARTKEY\n");
fprintf(fastjob,"   ,P_NAME\n");
fprintf(fastjob,"   ,P_MFGR\n");
fprintf(fastjob,"   ,P_BRAND\n");
fprintf(fastjob,"   ,P_TYPE\n");
fprintf(fastjob,"   ,P_SIZE\n");
fprintf(fastjob,"   ,P_CONTAINER\n");
fprintf(fastjob,"   ,P_RETAILPRICE\n");
fprintf(fastjob,"   ,P_COMMENT\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"VALUES\n");
fprintf(fastjob,"  (\n");
fprintf(fastjob,"   :F_P_PARTKEY,\n");
fprintf(fastjob,"   :F_P_NAME,\n");
fprintf(fastjob,"   :F_P_MFGR,\n");
fprintf(fastjob,"   :F_P_BRAND,\n");
fprintf(fastjob,"   :F_P_TYPE,\n");
fprintf(fastjob,"   :F_P_SIZE,\n");
fprintf(fastjob,"   :F_P_CONTAINER,\n");
fprintf(fastjob,"   :F_P_RETAILPRICE,\n");
fprintf(fastjob,"   :F_P_COMMENT\n");
break;
case PSUPP:
fprintf(fastjob,"    PS_PARTKEY\n");
fprintf(fastjob,"   ,PS_SUPPKEY\n");
fprintf(fastjob,"   ,PS_AVAILQTY\n");
fprintf(fastjob,"   ,PS_SUPPLYCOST\n");
fprintf(fastjob,"   ,PS_COMMENT\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"VALUES\n");
fprintf(fastjob,"  (\n");
fprintf(fastjob,"   :F_PS_PARTKEY,\n");
fprintf(fastjob,"   :F_PS_SUPPKEY,\n");
fprintf(fastjob,"   :F_PS_AVAILQTY,\n");
fprintf(fastjob,"   :F_PS_SUPPLYCOST,\n");
fprintf(fastjob,"   :F_PS_COMMENT\n");
break;
case REGION:
fprintf(fastjob,"    R_REGIONKEY\n");
fprintf(fastjob,"   ,R_NAME\n");
fprintf(fastjob,"   ,R_COMMENT\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"VALUES\n");
fprintf(fastjob,"  (\n");
fprintf(fastjob,"   :F_R_REGIONKEY,\n");
fprintf(fastjob,"   :F_R_NAME,\n");
fprintf(fastjob,"   :F_R_COMMENT\n");
break;
case SUPP:
fprintf(fastjob,"    S_SUPPKEY\n");
fprintf(fastjob,"   ,S_NAME\n");
fprintf(fastjob,"   ,S_ADDRESS\n");
fprintf(fastjob,"   ,S_NATIONKEY\n");
fprintf(fastjob,"   ,S_PHONE\n");
fprintf(fastjob,"   ,S_ACCTBAL\n");
fprintf(fastjob,"   ,S_COMMENT\n");
fprintf(fastjob,"  )\n");
fprintf(fastjob,"VALUES\n");
fprintf(fastjob,"  (\n");
fprintf(fastjob,"   :F_S_SUPPKEY,\n");
fprintf(fastjob,"   :F_S_NAME,\n");
fprintf(fastjob,"   :F_S_ADDRESS,\n");
fprintf(fastjob,"   :F_S_NATIONKEY,\n");
fprintf(fastjob,"   :F_S_PHONE,\n");
fprintf(fastjob,"   :F_S_ACCTBAL,\n");
fprintf(fastjob,"   :F_S_COMMENT\n");
```

```
break;
}

fprintf(fastjob,"  );\n");
fprintf(fastjob,"\n");
fprintf(fastjob,"END LOADING;\n");
fprintf(fastjob,"\n");
fprintf(fastjob,"LOGOFF;\n");

fflush(fastjob);

if (fclose(fastjob)!=0)
       fprintf(stderr, " Error closing fastjob file\n");
if (verbose)
fprintf(stderr," Closed %s and ready for fastload to read
it\n",fastname);

}

int
end_load(void)
{
#ifndef WIN32
       char temp[200];
#endif
       if (loadinprogress==0) return(0);

       if (loadinprogress==PART_PSUPP)
              {
#ifdef WIN32
                     FlushFileBuffers(pipehandle[PART]);
                     //DisconnectNamedPipe(pipehandle[PART]);
                     CloseHandle(pipehandle[PART]);
                     FlushFileBuffers(pipehandle[PSUPP]);
                     //DisconnectNamedPipe(pipehandle[PSUPP]);
                     CloseHandle(pipehandle[PSUPP]);

       WaitForSingleObject(processhandle[PSUPP],9999999);
#else
              fclose(pipefile[PART]);
                     fclose(pipefile[PSUPP]);
                     system("rm part.tbl.pipe");
                     system("rm partsupp.tbl.pipe");
#endif
                     fprintf(stderr," Sent all rows for part and
partsupp tables...\n");
              }
       else if (loadinprogress==ORDER_LINE)
              {
#ifdef WIN32
                     FlushFileBuffers(pipehandle[ORDER]);
                     //DisconnectNamedPipe(pipehandle[ORDER]);
                     CloseHandle(pipehandle[ORDER]);
                     FlushFileBuffers(pipehandle[LINE]);
                     //DisconnectNamedPipe(pipehandle[LINE]);
                     CloseHandle(pipehandle[LINE]);

       WaitForSingleObject(processhandle[LINE],9999999);
#else
              fclose(pipefile[ORDER]);
                     fclose(pipefile[LINE]);
                     system("rm order.tbl.pipe");
                     system("rm lineitem.tbl.pipe");
#endif
                     fprintf(stderr," Sent all rows for order and
lineitem tables...\n");
              }
       else
              {
#ifdef WIN32
                     FlushFileBuffers(pipehandle[loadinprogress]);

       //DisconnectNamedPipe(pipehandle[loadinprogress]);
                     CloseHandle(pipehandle[loadinprogress]);

       WaitForSingleObject(processhandle[loadinprogress],9999999);
#else
              fclose(pipefile[loadinprogress]);
                     strcpy(temp,"rm ");
                     strcat(temp,pipename);
                     system(temp);
#endif
                     if (loadinprogress==19)
                     {
                            fprintf(stderr," Sent all keys for
deletes ...\n");
                            system("rm deletes.fastload");
                     }
                     else
                            fprintf(stderr," Sent all rows for %s
...\n",tdefs[loadinprogress].name);
              }
       if (verbose)
              fprintf(stderr, " Closed the pipe...\n");
       loadinprogress = 0;
    return(0);
}



int
start_load2(int t,int mode)
{
       int rc;

       char temp[120];

#ifdef WIN32
       STARTUPINFO startinfo;
       PROCESS_INFORMATION procinfo;
#else
       pid_t fpid;
#endif

       pipename[0] = '\0';
#ifdef WIN32
```

```
        strcpy(pipename,"\\\\.\\pipe\\");
#endif
        if (t==19)
        {
                strcat(pipename,"deletes.pipe");
                strcpy(fastname,"deletes.fastload");
        }
        else
        {
                strcat(pipename,tdefs[t].name);
                strcat(pipename,".pipe");

                strcpy(fastname,tdefs[t].name);
                strcat(fastname,".fastload");
        }


        WriteFastloadJob(t,mode);

#ifdef WIN32
if (verbose)
        fprintf(stderr," Creating the named pipe... \n");

pipehandle[t] = CreateNamedPipe(pipename,
        PIPE_ACCESS_DUPLEX,
        PIPE_TYPE_BYTE | PIPE_READMODE_BYTE | PIPE_WAIT,
        3,
        32000,
        32000,
        1800000 /* 30 minutes */,NULL);

if (pipehandle[t]==INVALID_HANDLE_VALUE)
        {
        rc = GetLastError();
        fprintf(stderr," CreateNamedPipe error was %d\n",rc);
        exit(rc);
        }

fprintf(stderr," Spawning a copy of fastload...\n");


startinfo.cb = sizeof(STARTUPINFO);
startinfo.lpReserved = NULL;
startinfo.lpDesktop = NULL;
startinfo.lpTitle = NULL;
startinfo.dwFlags =0;
startinfo.cbReserved2 =0;
startinfo.lpReserved2 =0;
startinfo.hStdInput = NULL;
startinfo.hStdOutput = NULL;
startinfo.hStdError = NULL;

//if (! CreateProcess(NULL,"cmd /K fastload < custinmod.txt > test.out
2>&1",
//  NULL,NULL,FALSE,CREATE_NEW_CONSOLE | NORMAL_PRIORITY_CLASS,
//  NULL,NULL,&startinfo,&procinfo))
//      {
//      fprintf(stderr," Spawn of fastload failed,
rc=%d\n",GetLastError());
//      }

strcpy(temp,"cmd /K fastload.exe < ");
strcat(temp,fastname);
fprintf(stderr,"Running> %s\n",temp);

if (! CreateProcess(NULL,temp,
  NULL,NULL,FALSE, CREATE_NEW_CONSOLE|NORMAL_PRIORITY_CLASS,
  NULL,NULL,&startinfo,&procinfo))
        {
        rc = GetLastError();
        if (rc==ERROR_FILE_NOT_FOUND)
                fprintf(stderr," Spawn of fastload failed, file not
found\n");
        else
                fprintf(stderr," Spawn of fastload failed, rc=%d\n",rc);
        }
processhandle[t] = procinfo.hProcess;


fprintf(stderr," Waiting for Fastload to connect to the pipe... \n");
if (!ConnectNamedPipe(pipehandle[t],NULL))
        {
        rc = GetLastError();
        if (rc==ERROR_CALL_NOT_IMPLEMENTED)
                fprintf(stderr," Can't run this program on Win95 because
server side pipes don't exist\n");
        else
                fprintf(stderr," ConnectNamedPipe Error was %d\n",rc);
        exit(rc);
        }
fprintf(stderr," Pipe is connected... \n");

#else
        if (verbose)
                fprintf(stderr," Creating the named pipe\n");


        strcpy(temp,"mkfifo ");
        strcat(temp,pipename);
        if (verbose)
                fprintf(stderr,"%s\n",temp);
        rc= system(temp);
        if (rc!=0)
                {
                        if (rc==-1)
                                {
                                        fprintf(stderr," Error %d trying
to create the pipe\n",errno);
                                }
                        else
                                {
                                        if (rc!=256)
                                        fprintf(stderr," Create pipe
returned %d\n",rc);
                                }
```

```
                }

        if (verbose)
                fprintf(stderr," Spawning a copy of fastload for
%s...\n",fastname);
        if((fpid=fork())==0)
                {
                        strcpy(temp,"/usr/bin/fastload < ");
                        strcat(temp,fastname);
                        strcat(temp," > ");
                        strcat(temp,fastname);
                        strcat(temp,".out 2>&1");
                        if (verbose)
                                fprintf(stderr,"/usr/bin/sh -c
%s\n",temp);
                        execlp("/usr/bin/sh","/usr/bin/sh","-
c",temp,(char *)0);
                        fprintf(stderr, "spawn of fastload faild,
%d\n",errno);
                        fprintf(stderr, "You must start fastload
manually, using the following command:\n");
                        fprintf(stderr, "%s\n",temp);
                        exit(errno);
                }
        fastloadsstarted++;
        fprintf(stderr," Spawned a copy of fastload for %s, pid =
%d\n",fastname,fpid);

        if (verbose)
                fprintf(stderr," Connecting to the pipe for %s (%s)...
\n",fastname,pipename);
        pipefile[t] = fopen(pipename,"wb");
        if (pipefile[t]==NULL)
                {
                        rc = errno;
                        fprintf(stderr," Unable to open output file for
direct load\n");
                        fprintf(stderr," Error code = %d\n",rc);
                        exit(rc);
                }
        else
                {
                        if (verbose)
                                fprintf(stderr," Opened the pipe for
direct loading of %s\n",fastname);

                }

#endif
    return(0);
}

int
start_load(int t,int mode)
{

        if (loadinprogress!=0) end_load();
        if (t==PART_PSUPP)
                {
                        start_load2(PART,mode);
                        start_load2(PSUPP,mode);
                }
        else if (t==ORDER_LINE)
                {
                        start_load2(ORDER,mode);
                        start_load2(LINE,mode);
                }
        else
                start_load2(t,mode);

        loadinprogress = t;
        return(0);
}


/* First, some helper routines to copy data into the parcel we are
building */
static void DR_Strt()
{
        p1 = buffer + 2;       /* Skip 2 for the length field */
}
static void DR_End(int t)
{
        /* We must write to the pipe with a single I/O call, else we
           could get interleave problems when there are multiple
writers */
#ifdef WIN32
        unsigned long byteswritten;
#endif
        int len;
        int rc;
        reccnt++;

        *p1 = 0x0A; p1++;  /* Stick end-of-line in the buffer */
        len = (int)(p1-buffer);

        *((short *)buffer) =(short)(len-3);  /* Fill in length */

        #ifndef WIN32
        rc = fwrite(buffer,1,len,pipefile[t]);
        if (rc != len)
                {
                        fprintf(stderr, " Write to pipe or file failed,
errno=%d.\n",errno);
                        exit(1);
                }
        #else
        if (WriteFile(pipehandle[t],&buffer,len,&byteswritten,NULL))
                {
                        if (byteswritten!=(unsigned long)(len))
                                {
                                fprintf(stderr, " Byteswritten =
%d\n",byteswritten);
                                }

                }
```

```
        else
                {
                rc = GetLastError();
                if (rc==ERROR_BROKEN_PIPE)
                        fprintf(stderr, " Fastload broke the pipe
prematurely\n");
                else
                        fprintf(stderr, " unknown pipe error =
%d\n",rc);
                fprintf(stderr, " when trying to set row %d\n",reccnt);
                CloseHandle(pipehandle[t]);
                //fclose(pipefile);
                if (rc==0) rc++;
                exit(rc);
                }
        #endif
}

INLINE
static void DR_Int(long x)
{
        memcpy(p1,&x,4);
        p1 += 4;
}
INLINE
static void DR_Int64(long xlow, long xhigh)   /* Used only for
orderkey, which gets > 32 bits at times */
{
        double tempx;
        unsigned long templ;
        long temph;
        if (xhigh==0)
                {
                        memcpy(p1,&xlow,4);
                        p1 += 4;
                        memcpy(p1,&xhigh,4);
                        p1 += 4;
                }
        else
                {
                        /* Sigh, don't have an Int64 type, so need to
use double and convert to 2 longs */
                        tempx = xhigh*10000000.0+xlow;
                        temph = (long)(tempx/4294967296.0);
                        templ = (unsigned long)(tempx-
temph*4294967296.0);
                        memcpy(p1,&templ,4);
                        p1 += 4;
                        memcpy(p1,&temph,4);
                        p1 += 4;

                }

}
INLINE
static void DR_VStr(char * x)
{
        short len;
        len=(short)strlen(x);
        memcpy(p1,&len,2);
        p1 += 2;
        memcpy(p1,x,len);
        p1 += len;
}
INLINE
static void DR_Str(char * x,size_t y)
{
        memset(p1,' ',y);
        memcpy(p1,x,strlen(x));
        p1+=y;
}
INLINE
static void DR_Money(long x)
{
        long temp = 0;
        if (x<0)
                temp = -1;
        /*sprintf(p1, "%13.2f", x * 0.01);
        p1+=13;  */
        memcpy(p1,&x,4);
        p1 += 4;
        memcpy(p1,&temp,4);
        p1 += 4;

}
INLINE
static void DR_Char(char x)
{
        *p1 = x; p1++;
}
INLINE
static void DR_Date(char * x)
{
        long tempdate;
        int y,m,d;
        /*memset(p1,' ',10);
        memcpy(p1,x,strlen(x));
        p1+=10;*/
        y=atoi(x);
        m=atoi(x+5);
        d=atoi(x+8);
        if (y<1800 || y>2200 || m<1 || m>12 || d<1 || d>31)
                {
                fprintf(stderr," Help! I can't interpret the date
%s\n",x);
                exit(1);
                }
        tempdate=(y-1900)*10000+m*100+d;
        memcpy(p1,&tempdate,4);
        p1 += 4;

}

/*now the routines from load_stub.c */

int
```

```
close_direct(void)
{
#ifndef WIN32
        int c;
        int i;
        int status;
#endif
        if (loadinprogress!=0) end_load();

        loadinprogress = 0;

#ifndef WIN32
        c = fastloadsstarted;
        fprintf(stderr," Finished sending all rows to all tables.\n");
        fprintf(stderr," Now waiting for all fastloads to end...\n");
    while (c)
        {
        i = wait(&status);
        if (i == -1 && fastloadsstarted)
            {
            fprintf(stderr, "We lost one of the fastloads?\n");
            return(-2);
            }
        if (status & 0xFF)
            {
            if (status & 0xFF == 0117)
                fprintf(stderr,"Fastload Process %d: STOPPED\n", i);
            else
                fprintf(stderr,"Fastload Process %d: rcvd signal %d\n",
                    i, status&0x7F);
            }
        c--;
        }
#else
        Sleep(1000);
#endif
#ifdef WIN32
        system("erase *.tbl.pipe");
        system("erase *.tbl.fastload");
#else
        /*system("rm *.tbl.pipe");     */
        system("rm *.tbl.fastload");
#endif
        fprintf(stderr," **** Done with direct loading ****\n");
    return(0);
}

int
prep_direct(char * dbnamestr)
{

        fastloadsstarted = 0;
        strcpy(logonstr,dbnamestr);
        if (strcmp(logonstr,"dss")==0 || strcmp(logonstr,"")==0)
                {
                        fprintf(stderr," You MUST use the -n option to
define the Teradata logon string\n");
                        exit(1);
                }
        if (verbose)
                fprintf(stderr," Teradata logon is %s\n",dbnamestr);
        if (loadinprogress!=0) end_load();

    /* any preload prep goes here */
    return(0);
}

int
hd_cust (FILE *f)
{
    static int count = 0;
        if (f);

    if (! count++)
        printf("No header has been defined for the customer table\n");

    return(0);
}

/*
CREATE TABLE TPCD.CUSTOMER ( C_CUSTKEY     INTEGER NOT NULL,
                             C_NAME        VARCHAR(25) NOT NULL,
                             C_ADDRESS     VARCHAR(40) NOT NULL,
                             C_NATIONKEY   INTEGER NOT NULL,
                             C_PHONE       CHAR(15) NOT NULL,
                             C_ACCTBAL     DECIMAL(15,2)   NOT NULL,
                             C_MKTSEGMENT  CHAR(10) NOT NULL,
                             C_COMMENT     VARCHAR(117) NOT NULL);
*/

int
ld_cust (customer_t *cp, int mode)
{
        if(mode);
        if (loadinprogress!=CUST) start_load(CUST,mode);
        DR_Strt();
        DR_Int(cp->custkey);
    DR_VStr(cp->name);
        DR_VStr(cp->address);
        DR_Int(cp->nation_code);
        DR_Str(cp->phone,PHONE_LEN);
        DR_Money(cp->acctbal);
        DR_Str(cp->mktsegment,10);
        DR_VStr(cp->comment);
        DR_End(CUST);
        return(0);
}


int
hd_part (FILE *f)
{
    static int count = 0;
        if(f);

    if (! count++)
```

```
        printf("No header has been defined for the part table\n");

    return(0);
}

/*
CREATE TABLE TPCD.PART  ( P_PARTKEY     INTEGER NOT NULL,
                          P_NAME        VARCHAR(55) NOT NULL,
                          P_MFGR        CHAR(25) NOT NULL,
                          P_BRAND       CHAR(10) NOT NULL,
                          P_TYPE        VARCHAR(25) NOT NULL,
                          P_SIZE        INTEGER NOT NULL,
                          P_CONTAINER   CHAR(10) NOT NULL,
                          P_RETAILPRICE DECIMAL(15,2) NOT NULL,
                          P_COMMENT     VARCHAR(23) NOT NULL );
*/
int
ld_part (part_t *pp, int mode)
{

        if(mode);
        if (loadinprogress!=PART && loadinprogress!=PART_PSUPP)
start_load(PART,mode);
        DR_Strt();
        DR_Int(pp->partkey);
        DR_VStr(pp->name);
        DR_Str(pp->mfgr,P_MFG_LEN);
        DR_Str(pp->brand,P_BRND_LEN);
        DR_VStr(pp->type);
        DR_Int(pp->size);
        DR_Str(pp->container,P_CNTR_LEN);
        DR_Money(pp->retailprice);
        DR_VStr(pp->comment);
        DR_End(PART);
        return(0);

}

/*
CREATE TABLE TPCD.PARTSUPP ( PS_PARTKEY     INTEGER NOT NULL,
                             PS_SUPPKEY     INTEGER NOT NULL,
                             PS_AVAILQTY    INTEGER NOT NULL,
                             PS_SUPPLYCOST  DECIMAL(15,2)  NOT NULL,
                             PS_COMMENT     VARCHAR(199) NOT NULL );
*/

int
ld_psupp (part_t *pp, int mode)
{
        int i;
        if(mode);
        if (loadinprogress!=PSUPP && loadinprogress!=PART_PSUPP)
start_load(PSUPP,mode);
        for (i = 0; i < SUPP_PER_PART; i++)
      {
              DR_Strt();
              DR_Int(pp->s[i].partkey);
              DR_Int(pp->s[i].suppkey);
              DR_Int(pp->s[i].qty);
              DR_Money(pp->s[i].scost);
              DR_VStr(pp->s[i].comment);
              DR_End(PSUPP);
      }
        return(0);

}

int
hd_time (FILE *f)
{
    static int count = 0;
       if(f);

    if (! count++)
        printf("No header has been defined for the time table\n");

    return(0);
}

/*
CREATE TABLE TPCD.TIME1    ( T_TIMEKEY     INTEGER NOT NULL,
--                           T_ALPHA       DATE NOT NULL,
--                           T_YEAR        INTEGER NOT NULL,
--                           T_MONTH       INTEGER NOT NULL,
--                           T_WEEK        INTEGER NOT NULL,
--                           T_DAY         INTEGER NOT NULL );
*/

int
ld_time (dss_time_t *tp, int mode)
{
    if(mode);

        DR_Strt();
        DR_Int(tp->timekey);
    DR_Date(tp->alpha);
    DR_Int(tp->year);
    DR_Int(tp->month);
    DR_Int(tp->week);
    DR_Int(tp->day);
        DR_End(TIME);

    return(0);
}


int
hd_supp (FILE *f)
{
    static int count = 0;
       if(f);

    if (! count++)
        printf("No header has been defined for the supplier table\n");

    return(0);
```

```
}

/*
CREATE TABLE TPCD.SUPPLIER ( S_SUPPKEY     INTEGER NOT NULL,
                             S_NAME        CHAR(25) NOT NULL,
                             S_ADDRESS     VARCHAR(40) NOT NULL,
                             S_NATIONKEY   INTEGER NOT NULL,
                             S_PHONE       CHAR(15) NOT NULL,
                             S_ACCTBAL     DECIMAL(15,2) NOT NULL,
                             S_COMMENT     VARCHAR(101) NOT NULL);
*/

int
ld_supp (supplier_t *sp, int mode)
{
        if(mode);
        if (loadinprogress!=SUPP) start_load(SUPP,mode);
        DR_Strt();
        DR_Int(sp->suppkey);
    DR_Str(sp->name,25);
        DR_VStr(sp->address);
        DR_Int(sp->nation_code);
        DR_Str(sp->phone,15);
        DR_Money(sp->acctbal);
        DR_VStr(sp->comment);
        DR_End(SUPP);
    return(0);
}


int
hd_order (FILE *f)
{
    static int count = 0;
        if(f);

    if (! count++)
        printf("No header has been defined for the order table\n");

    return(0);
}

/*
CREATE TABLE TPCD.ORDER  ( O_ORDERKEY       INTEGER NOT NULL,
                           O_CUSTKEY        INTEGER NOT NULL,
                           O_ORDERSTATUS    CHAR(1) NOT NULL,
                           O_TOTALPRICE     DECIMAL(15,2) NOT NULL,
                           O_ORDERDATE      DATE NOT NULL,
                           O_ORDERPRIORITY  CHAR(15) NOT NULL,  -- R
                           O_CLERK          CHAR(15) NOT NULL,  -- R
                           O_SHIPPRIORITY   INTEGER NOT NULL,
                           O_COMMENT        VARCHAR(79) NOT NULL);
*/

int
ld_order (order_t *p, int mode)
{
        if(mode);
        if (loadinprogress!=ORDER && loadinprogress!=ORDER_LINE)
start_load(ORDER,mode);
        DR_Strt();
        if (scale>300)
                {
                        DR_Int64(p->okey_low,p->okey_high);
                }
        else
                {
                        if (p->okey_high==0)
                        DR_Int(p->okey_low);
                        else
                                DR_Int(p->okey_high*10000000+p-
>okey_low);
                }
        DR_Int(p->custkey);
        DR_Char(p->orderstatus);
        DR_Money(p->totalprice);
        DR_Date(p->odate);
        DR_Str(p->opriority,O_OPRIO_LEN);
        DR_Str(p->clerk,O_CLRK_LEN);
        DR_Int(p->spriority);
#ifdef _WIN32
        if (strlen(p->comment)<0||strlen(p->comment)>79)
                __asm int 3;
#endif
        DR_VStr(p->comment);
        DR_End(ORDER);
    return(0);
}

/*
CREATE TABLE TPCD.LINEITEM ( L_ORDERKEY    INTEGER NOT NULL,
                             L_PARTKEY     INTEGER NOT NULL,
                             L_SUPPKEY     INTEGER NOT NULL,
                             L_LINENUMBER  INTEGER NOT NULL,
                             L_QUANTITY    DECIMAL(15,2) NOT NULL,
                             L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL,
                             L_DISCOUNT    DECIMAL(15,2) NOT NULL,
                             L_TAX         DECIMAL(15,2) NOT NULL,
                             L_RETURNFLAG  CHAR(1) NOT NULL,
                             L_LINESTATUS  CHAR(1) NOT NULL,
                             L_SHIPDATE    DATE NOT NULL,
                             L_COMMITDATE  DATE NOT NULL,
                             L_RECEIPTDATE DATE NOT NULL,
                             L_SHIPINSTRUCT CHAR(25) NOT NULL,  -- R
                             L_SHIPMODE     CHAR(10) NOT NULL,  -- R
                             L_COMMENT      VARCHAR(44) NOT NULL);
*/
ld_line (order_t *p, int mode)
{
        int i;
    if(mode);
        if (loadinprogress!=LINE && loadinprogress!=ORDER_LINE)
start_load(LINE,mode);
        for (i = 0; i < p->lines; i++)
        {
        DR_Strt();
```

```
        if (scale>300)
                {
                        DR_Int64(p->l[i].okey_low,p->l[i].okey_high);
                }
        else
                {
                        if (p->l[i].okey_high==0)
                        DR_Int(p->l[i].okey_low);
                        else
                                DR_Int(p->l[i].okey_high*10000000+p-
>l[i].okey_low);
                }
        DR_Int(p->l[i].partkey);
        DR_Int(p->l[i].suppkey);
        DR_Int(p->l[i].lcnt);
        DR_Int(p->l[i].quantity);
        DR_Money(p->l[i].eprice);
        DR_Money(p->l[i].discount);
        DR_Money(p->l[i].tax);
        DR_Char(p->l[i].rflag[0]);
        DR_Char(p->l[i].lstatus[0]);
        DR_Date(p->l[i].sdate);
        DR_Date(p->l[i].cdate);
        DR_Date(p->l[i].rdate);
        DR_Str(p->l[i].shipinstruct, L_INST_LEN);
        DR_Str(p->l[i].shipmode, L_SMODE_LEN);
        DR_VStr(p->l[i].comment);
        DR_End(LINE);
        }
    return(0);
}



int
hd_psupp (FILE *f)
{
    static int count = 0;
        if(f);

    if (! count++)
        printf("%s %s\n",
            "No header has been defined for the",
            "part supplier table");

    return(0);
}


int
hd_line (FILE *f)
{
    static int count = 0;
        if(f);

    if (! count++)
        printf("No header has been defined for the lineitem table\n");

    return(0);
}

int
hd_nation (FILE *f)
{
    static int count = 0;
        if(f);

    if (! count++)
        printf("No header has been defined for the nation table\n");

    return(0);
}

/*
CREATE TABLE TPCD.NATION  ( N_NATIONKEY  INTEGER NOT NULL,
                            N_NAME       CHAR(25) NOT NULL,
                            N_REGIONKEY  INTEGER NOT NULL,
                            N_COMMENT    VARCHAR(152));
*/

int
ld_nation (code_t *cp, int mode)
{
        if(mode);
        if (loadinprogress!=NATION) start_load(NATION,mode);
        DR_Strt();
        DR_Int(cp->code);
    DR_Str(cp->text,NATION_LEN);
        DR_Int(cp->join);
        DR_VStr(cp->comment);
        DR_End(NATION);
        return(0);
}

int
hd_region (FILE *f)
{
    static int count = 0;
        if(f);

    if (! count++)
        printf("No header has been defined for the region table\n");

    return(0);
}

/*
CREATE TABLE TPCD.REGION  ( R_REGIONKEY  INTEGER NOT NULL,
                            R_NAME       CHAR(25) NOT NULL,
                            R_COMMENT    VARCHAR(152));
*/

int
ld_region (code_t *cp, int mode)
{
        if(mode);
```

```
        if (loadinprogress!=REGION) start_load(REGION,mode);
        DR_Strt();
        DR_Int(cp->code);
        DR_Str(cp->text,REGION_LEN);
        DR_VStr(cp->comment);
        DR_End(REGION);

    return(0);
}

int
ld_order_line (order_t *p, int mode)
{
        if (loadinprogress!=ORDER_LINE) start_load(ORDER_LINE,mode);

    ld_order(p, mode);
    ld_line (p, mode);

    return(0);
}

int
hd_order_line (FILE *f)
{

    hd_order(f);
    hd_line (f);

    return(0);
}

int
ld_part_psupp (part_t *p, int mode)
{
        if (loadinprogress!=PART_PSUPP) start_load(PART_PSUPP,mode);
    ld_part(p, mode);
    ld_psupp (p, mode);

    return(0);
}

int
hd_part_psupp (FILE *f)
{
    hd_part(f);
    hd_psupp(f);

    return(0);
}

int
ld_deletes (long okey, int mode)
{
        if(mode);
        if (loadinprogress!=19) start_load(19,mode);
        DR_Strt();
        if (scale>300)
                {
                        DR_Int64(okey,0);
                }
        else
                {
                        DR_Int(okey);
                }

        DR_End(19);
    return(0);
}
```

F.3 Load Scripts

```
#!/bin/sh
#
#
#   Variables describing current system
#
system=local
SF=100
logon="${system}/tpcd100g,tpcd100g"
Logon=".logon $logon"
dbclogon=${system}/dbc,dbc
D=/home2/tpcd
#
#

ulimit -n 1024
ulimit -f unlimited
ulimit -v unlimited
ulimit -a

#
#   Setup any database specific controls
#
echo `date``:  Setting Cylinder Fill Factor to 100%'
/tpasw/bin/dbscontrol << INS0
modify fi 1 = 0
write
quit
INS0

cd $D/pfast/master
echo "LOGON $logon;" > LOGON

echo `date``:  Starting load of lineitem table...pid=$$...'
./Build lineitem
./fastload < inmod.lineitem > inmod.lineitem.out 2>&1
$D/bin/pagefrank "lineitem table load complete"

echo `date``:  Starting load of order table...'
./Build order
./fastload < inmod.order > inmod.order.out 2>&1
```

```
 echo 'date'':  Starting load of partsupp table...'
./Build partsupp
./fastload < inmod.partsupp > inmod.partsupp.out 2>&1

echo 'date'':  Starting load of part table...'
./Build part
./fastload < inmod.part > inmod.part.out 2>&1

echo 'date'':  Starting load of supplier table...'
./Build supplier
./fastload < inmod.supplier > inmod.supplier.out 2>&1

echo 'date'':  Starting load of customer table'
./Build customer
./fastload < inmod.customer > inmod.customer.out 2>&1

echo 'date'':  Starting load of region table'
./fastload < inmod.region > inmod.region.out 2>&1

echo 'date'':  Starting load of nation table'
./fastload < inmod.nation > inmod.nation.out 2>&1
echo 'date'':  TPC-D table load complete'
$D/bin/pagefrank "table load complete"

echo 'date'':  Starting prep job...'
cd $D/preps
echo $Logon | cat - prep_bench | bteq > prep_bench.out 2>&1
echo $Logon | cat - prep_j4j9 | bteq > prep_j4j9.out 2>&1

echo 'date'':  preps Complete.'

$D/bin/pagefrank "preps complete"

echo 'date'':  Starting audit script ...'
cd $D/audit
echo $Logon | cat - table_contents | bteq  >
table_contents.${SF}Gload.out 2> table_contents.${SF}Gload.err
echo $Logon | cat - audit.sql | bteq > audit.sql.${SF}Gload.out 2>
audit.sql.${SF}Gload.err
echo 'date'':  Select Count and help_all queries complete'

$D/bin/pagefrank "audit complete"


banner "TPC-D Database" "  Load" " Complete" | wall
echo 'date'':  TPC-D Load complete'
echo 'date'':  EXITING'
```

Text of Build Script

```
#!/bin/sh
#
if [ $# != 1 ]; then
        echo "usage: $0 <TPC-D table name>" 1>&2
        exit 1
fi
tbl=$1
DFLTSESS=384
#
#  Get Common REMOTE commands from master directory
#  Modify to apply to particular table we are doing
#
sed "s/{TBL}/$1/g" REMOTES > R.tmp
#
#  If master inmod script has a session declaration
#  use it, otherwise use default defined above
#
egrep -i SESSION MASTER/inmod.${tbl} > S.tmp
if [ $? -ne 0 ]; then
  echo "SESSIONS ${DFLTSESS};\n" > S.tmp
fi
#
#  Put the new script together
#  SESSIONS,REMOTE, LOGON,INMOD (sans SESSION)
#
cat S.tmp R.tmp LOGON > inmod.${tbl}
#
#  add all of the master script sans any SESSION
#  statements
egrep -v -i SESSION MASTER/inmod.${tbl} >> inmod.${tbl}
#
#  Finally get rid of the tmp files
rm -f S.tmp R.tmp
```

INMOD FILES CREATED BY THE Build SCRIPT

---

inmod.lineitem

```
SESSIONS 36;

REMOTE dino4 root /home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE dino4 root /home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE dino4 root /home2/tpcd/pfast/inmod/lineitem2/lineitem;
REMOTE dino4 root /home2/tpcd/pfast/inmod/lineitem3/lineitem;

LOGON local/tpcd100g,tpcd100g;

DROP TABLE LINEITEM;
DROP TABLE ERRLINEITEM1;
DROP TABLE ERRLINEITEM2;

CREATE MULTISET TABLE LINEITEM, DATABLOCKSIZE=28.5 KILOBYTES, AFTER
JOURNAL
  (
    L_ORDERKEY       INTEGER not null
   ,L_PARTKEY        INTEGER not null
   ,L_SUPPKEY        INTEGER not null
   ,L_LINENUMBER     INTEGER not null
   ,L_QUANTITY       DECIMAL(15,2) not null
   ,L_EXTENDEDPRICE  DECIMAL(15,2) not null
   ,L_DISCOUNT       DECIMAL(15,2) not null
   ,L_TAX            DECIMAL(15,2) not null
```

```
   ,L_RETURNFLAG     VARCHAR(1) CASESPECIFIC not null
   ,L_LINESTATUS     VARCHAR(1) CASESPECIFIC not null
   ,L_SHIPDATE       DATE FORMAT 'yyyy-mm-dd' not null
   ,L_COMMITDATE     DATE FORMAT 'yyyy-mm-dd' not null
   ,L_RECEIPTDATE    DATE FORMAT 'yyyy-mm-dd' not null
   ,L_SHIPINSTRUCT   VARCHAR(25) CASESPECIFIC not null
   ,L_SHIPMODE       VARCHAR(10) CASESPECIFIC not null
   ,L_COMMENT        VARCHAR(44) CASESPECIFIC not null
  )
;

BEGIN LOADING LINEITEM
      ERRORFILES ERRLINEITEM1, ERRLINEITEM2
      CHECKPOINT 300;

DEFINE
    F_L_ORDERKEY       (INTEGER)
   ,F_L_PARTKEY        (INTEGER)
   ,F_L_SUPPKEY        (INTEGER)
   ,F_L_LINENUMBER     (INTEGER)
   ,F_L_QUANTITY       (INTEGER)
   ,F_L_EXTENDEDPRICE  (DECIMAL(15,2))
   ,F_L_DISCOUNT       (DECIMAL(15,2))
   ,F_L_TAX            (DECIMAL(15,2))
   ,F_L_RETURNFLAG     (CHAR(1))
   ,F_L_LINESTATUS     (CHAR(1))
   ,F_L_SHIPDATE       (DATE)
   ,F_L_COMMITDATE     (DATE)
   ,F_L_RECEIPTDATE    (DATE)
   ,F_L_SHIPINSTRUCT   (CHAR(25))
   ,F_L_SHIPMODE       (CHAR(10))
   ,F_L_COMMENT        (VARCHAR(44))
INMOD=BLKEXIT;

SHOW;

INSERT INTO LINEITEM
  (
    L_ORDERKEY
   ,L_PARTKEY
   ,L_SUPPKEY
   ,L_LINENUMBER
   ,L_QUANTITY
   ,L_EXTENDEDPRICE
   ,L_DISCOUNT
   ,L_TAX
   ,L_RETURNFLAG
   ,L_LINESTATUS
   ,L_SHIPDATE
   ,L_COMMITDATE
   ,L_RECEIPTDATE
   ,L_SHIPINSTRUCT
   ,L_SHIPMODE
   ,L_COMMENT
  )
VALUES
  (
   :F_L_ORDERKEY,
   :F_L_PARTKEY,
   :F_L_SUPPKEY,
   :F_L_LINENUMBER,
   :F_L_QUANTITY,
   :F_L_EXTENDEDPRICE,
   :F_L_DISCOUNT,
   :F_L_TAX,
   :F_L_RETURNFLAG,
   :F_L_LINESTATUS,
   :F_L_SHIPDATE,
   :F_L_COMMITDATE,
   :F_L_RECEIPTDATE,
   :F_L_SHIPINSTRUCT,
   :F_L_SHIPMODE,
   :F_L_COMMENT
  );

END LOADING;

LOGOFF;
---------------------------------------
inmod.order
---------------------------------------
SESSIONS 36;

REMOTE dino4 root /home2/tpcd/pfast/inmod/order/order;
REMOTE dino4 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE dino4 root /home2/tpcd/pfast/inmod/order2/order;
REMOTE dino4 root /home2/tpcd/pfast/inmod/order3/order;

LOGON local/tpcd100g,tpcd100g;

DROP TABLE ORDERTBL;
DROP TABLE ERRORDERTBL1;
DROP TABLE ERRORDERTBL2;

CREATE MULTISET TABLE ORDERTBL, DATABLOCKSIZE=29.5 KILOBYTES, AFTER
JOURNAL
  (
    O_ORDERKEY       INTEGER not null
   ,O_CUSTKEY        INTEGER not null
   ,O_ORDERSTATUS    VARCHAR(1) CASESPECIFIC not null
   ,O_TOTALPRICE     DECIMAL(15,2) not null
   ,O_ORDERDATE      DATE FORMAT 'yyyy-mm-dd' not null
   ,O_ORDERPRIORITY  VARCHAR(15) CASESPECIFIC not null
   ,O_CLERK          VARCHAR(15) CASESPECIFIC not null
   ,O_SHIPPRIORITY   INTEGER not null
   ,O_COMMENT        VARCHAR(79) CASESPECIFIC not null
  )
UNIQUE PRIMARY INDEX( O_ORDERKEY );

BEGIN LOADING ORDERTBL
      ERRORFILES ERRORDERTBL1, ERRORDERTBL2;

DEFINE
    F_O_ORDERKEY       (INTEGER)
   ,F_O_CUSTKEY        (INTEGER)
   ,F_O_ORDERSTATUS    (CHAR(1))
```

```
   ,F_O_TOTALPRICE     (DECIMAL(15,2))
   ,F_O_ORDERDATE      (DATE)
   ,F_O_ORDERPRIORITY  (CHAR(15))
   ,F_O_CLERK          (CHAR(15))
   ,F_O_SHIPPRIORITY   (INTEGER)
   ,F_O_COMMENT        (VARCHAR(79))
INMOD=BLKEXIT;

SHOW;

INSERT INTO ORDERTBL
  (
    O_ORDERKEY
   ,O_CUSTKEY
   ,O_ORDERSTATUS
   ,O_TOTALPRICE
   ,O_ORDERDATE
   ,O_ORDERPRIORITY
   ,O_CLERK
   ,O_SHIPPRIORITY
   ,O_COMMENT
  )
VALUES
  (
   :F_O_ORDERKEY,
   :F_O_CUSTKEY,
   :F_O_ORDERSTATUS,
   :F_O_TOTALPRICE,
   :F_O_ORDERDATE,
   :F_O_ORDERPRIORITY,
   :F_O_CLERK,
   :F_O_SHIPPRIORITY,
   :F_O_COMMENT
  );

END LOADING;

LOGOFF;
---------------------------------------
inmod.partsupp
---------------------------------------

SESSIONS 36;

REMOTE dino4 root /home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE dino4 root /home2/tpcd/pfast/inmod/partsupp1/partsupp;
REMOTE dino4 root /home2/tpcd/pfast/inmod/partsupp2/partsupp;
REMOTE dino4 root /home2/tpcd/pfast/inmod/partsupp3/partsupp;

LOGON local/tpcd100g,tpcd100g;

DROP TABLE PARTSUPP;
DROP TABLE ERRPARTSUPP1;
DROP TABLE ERRPARTSUPP2;

CREATE MULTISET TABLE PARTSUPP
  (
    PS_PARTKEY     INTEGER not null
   ,PS_SUPPKEY     INTEGER not null
   ,PS_AVAILQTY    INTEGER not null
   ,PS_SUPPLYCOST  DECIMAL(15,2) not null
   ,PS_COMMENT     VARCHAR(199) CASESPECIFIC not null
  )
;

BEGIN LOADING PARTSUPP
      ERRORFILES ERRPARTSUPP1, ERRPARTSUPP2;

DEFINE
    F_PS_PARTKEY     (INTEGER)
   ,F_PS_SUPPKEY     (INTEGER)
   ,F_PS_AVAILQTY    (INTEGER)
   ,F_PS_SUPPLYCOST  (DECIMAL(15,2))
   ,F_PS_COMMENT     (VARCHAR(199))
INMOD=BLKEXIT;

SHOW;

INSERT INTO PARTSUPP
  (
    PS_PARTKEY
   ,PS_SUPPKEY
   ,PS_AVAILQTY
   ,PS_SUPPLYCOST
   ,PS_COMMENT
  )
VALUES
  (
   :F_PS_PARTKEY,
   :F_PS_SUPPKEY,
   :F_PS_AVAILQTY,
   :F_PS_SUPPLYCOST,
   :F_PS_COMMENT
  );

END LOADING;

LOGOFF;

---------------------------------------
inmod.part
---------------------------------------

SESSIONS 36;

REMOTE dino4 root /home2/tpcd/pfast/inmod/part/part;
REMOTE dino4 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE dino4 root /home2/tpcd/pfast/inmod/part2/part;
REMOTE dino4 root /home2/tpcd/pfast/inmod/part3/part;

LOGON local/tpcd100g,tpcd100g;

DROP TABLE PARTTBL;
DROP TABLE ERRPARTTBL1;
DROP TABLE ERRPARTTBL2;
```

```
CREATE MULTISET TABLE PARTTBL
  (
    P_PARTKEY      INTEGER not null
   ,P_NAME         VARCHAR(55) CASESPECIFIC not null
   ,P_MFGR         VARCHAR(25) CASESPECIFIC not null
   ,P_BRAND        VARCHAR(10) CASESPECIFIC not null
   ,P_TYPE         VARCHAR(25) CASESPECIFIC not null
   ,P_SIZE         INTEGER not null
   ,P_CONTAINER    VARCHAR(10) CASESPECIFIC not null
   ,P_RETAILPRICE  DECIMAL(15,2) not null
   ,P_COMMENT      VARCHAR(23) CASESPECIFIC not null
  )
UNIQUE PRIMARY INDEX( P_PARTKEY );

BEGIN LOADING PARTTBL
      ERRORFILES ERRPARTTBL1, ERRPARTTBL2;

DEFINE
    F_P_PARTKEY      (INTEGER)
   ,F_P_NAME         (VARCHAR(55))
   ,F_P_MFGR         (CHAR(25))
   ,F_P_BRAND        (CHAR(10))
   ,F_P_TYPE         (VARCHAR(25))
   ,F_P_SIZE         (INTEGER)
   ,F_P_CONTAINER    (CHAR(10))
   ,F_P_RETAILPRICE  (DECIMAL(15,2))
   ,F_P_COMMENT      (VARCHAR(23))
INMOD=BLKEXIT;

SHOW;

INSERT INTO PARTTBL
  (
    P_PARTKEY
   ,P_NAME
   ,P_MFGR
   ,P_BRAND
   ,P_TYPE
   ,P_SIZE
   ,P_CONTAINER
   ,P_RETAILPRICE
   ,P_COMMENT
  )
VALUES
  (
   :F_P_PARTKEY,
   :F_P_NAME,
   :F_P_MFGR,
   :F_P_BRAND,
   :F_P_TYPE,
   :F_P_SIZE,
   :F_P_CONTAINER,
   :F_P_RETAILPRICE,
   :F_P_COMMENT
  );

END LOADING;

LOGOFF;
---------------------------------------
inmod.customer
---------------------------------------

SESSIONS 36;

REMOTE dino4 root /home2/tpcd/pfast/inmod/customer/customer;
REMOTE dino4 root /home2/tpcd/pfast/inmod/customer1/customer;
REMOTE dino4 root /home2/tpcd/pfast/inmod/customer2/customer;
REMOTE dino4 root /home2/tpcd/pfast/inmod/customer3/customer;

LOGON local/tpcd100g,tpcd100g;

DROP TABLE CUSTOMER;
DROP TABLE ERRCUSTOMER1;
DROP TABLE ERRCUSTOMER2;

CREATE MULTISET TABLE CUSTOMER
  (
    C_CUSTKEY     INTEGER not null
   ,C_NAME        VARCHAR(25) CASESPECIFIC not null
   ,C_ADDRESS     VARCHAR(40) CASESPECIFIC not null
   ,C_NATIONKEY   INTEGER not null
   ,C_PHONE       VARCHAR(15) CASESPECIFIC not null
   ,C_ACCTBAL     DECIMAL(15,2) not null
   ,C_MKTSEGMENT  VARCHAR(10) CASESPECIFIC not null
   ,C_COMMENT     VARCHAR(117) CASESPECIFIC not null
  )
UNIQUE PRIMARY INDEX( C_CUSTKEY );

BEGIN LOADING CUSTOMER
      ERRORFILES ERRCUSTOMER1, ERRCUSTOMER2;

DEFINE
   F_C_CUSTKEY     (INTEGER)
  ,F_C_NAME        (VARCHAR(25))
  ,F_C_ADDRESS     (VARCHAR(40))
  ,F_C_NATIONKEY   (INTEGER)
  ,F_C_PHONE       (CHAR(15))
  ,F_C_ACCTBAL     (DECIMAL(15,2))
  ,F_C_MKTSEGMENT  (CHAR(10))
  ,F_C_COMMENT     (VARCHAR(117))
INMOD=BLKEXIT;

SHOW;

INSERT INTO CUSTOMER
  (
    C_CUSTKEY
   ,C_NAME
   ,C_ADDRESS
   ,C_NATIONKEY
   ,C_PHONE
   ,C_ACCTBAL
   ,C_MKTSEGMENT
   ,C_COMMENT
  )
```

```
VALUES
  (
   :F_C_CUSTKEY,
   :F_C_NAME,
   :F_C_ADDRESS,
   :F_C_NATIONKEY,
   :F_C_PHONE,
   :F_C_ACCTBAL,
   :F_C_MKTSEGMENT,
   :F_C_COMMENT
  );

END LOADING;

LOGOFF;
---------------------------------------
inmod.supplier
---------------------------------------


SESSIONS 36;

REMOTE dino4 root /home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE dino4 root /home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE dino4 root /home2/tpcd/pfast/inmod/supplier2/supplier;
REMOTE dino4 root /home2/tpcd/pfast/inmod/supplier3/supplier;

LOGON local/tpcd100g,tpcd100g;

DROP TABLE SUPPLIER;
DROP TABLE ERRSUPPLIER1;
DROP TABLE ERRSUPPLIER2;

CREATE MULTISET TABLE SUPPLIER
  (
    S_SUPPKEY    INTEGER not null
   ,S_NAME       VARCHAR(25) CASESPECIFIC not null
   ,S_ADDRESS    VARCHAR(40) CASESPECIFIC not null
   ,S_NATIONKEY  INTEGER not null
   ,S_PHONE      VARCHAR(15) CASESPECIFIC not null
   ,S_ACCTBAL    DECIMAL(15,2) not null
   ,S_COMMENT    VARCHAR(101) CASESPECIFIC not null
  )
UNIQUE PRIMARY INDEX( S_SUPPKEY );

BEGIN LOADING SUPPLIER
      ERRORFILES ERRSUPPLIER1, ERRSUPPLIER2;

DEFINE
   F_S_SUPPKEY    (INTEGER)
  ,F_S_NAME       (CHAR(25))
  ,F_S_ADDRESS    (VARCHAR(40))
  ,F_S_NATIONKEY  (INTEGER)
  ,F_S_PHONE      (CHAR(15))
  ,F_S_ACCTBAL    (DECIMAL(15,2))
  ,F_S_COMMENT    (VARCHAR(101))
INMOD=BLKEXIT;

SHOW;

INSERT INTO SUPPLIER
  (
    S_SUPPKEY
   ,S_NAME
   ,S_ADDRESS
   ,S_NATIONKEY
   ,S_PHONE
   ,S_ACCTBAL
   ,S_COMMENT
  )
VALUES
  (
   :F_S_SUPPKEY,
   :F_S_NAME,
   :F_S_ADDRESS,
   :F_S_NATIONKEY,
   :F_S_PHONE,
   :F_S_ACCTBAL,
   :F_S_COMMENT
  );

END LOADING;

LOGOFF;

---------------------------------------
inmod.nation
---------------------------------------

SESSIONS 1;
REMOTE dino4 root /home2/tpcd/pfast/inmod/nation/nation;

LOGON local/tpcd100g,tpcd100g;

DROP TABLE NATION;
DROP TABLE ERRNATIONTMP1;
DROP TABLE ERRNATIONTMP2;

CREATE MULTISET TABLE NATION
  (
   N_NATIONKEY INTEGER NOT NULL
  ,N_NAME VARCHAR(25) CASESPECIFIC NOT NULL
  ,N_REGIONKEY INTEGER NOT NULL
  ,N_COMMENT VARCHAR(152) CASESPECIFIC NOT NULL
  )
PRIMARY INDEX ( N_NAME );

BEGIN LOADING NATION
      ERRORFILES ERRNATIONTMP1, ERRNATIONTMP2;

DEFINE
   F_N_NATIONKEY (INTEGER)
  ,F_N_NAME (CHAR(25))
  ,F_N_REGIONKEY (INTEGER)
  ,F_N_COMMENT (VARCHAR(152))
INMOD=BLKEXIT;
```

```
SHOW;

INSERT INTO NATION
 (
   N_NATIONKEY
  ,N_NAME
  ,N_REGIONKEY
  ,N_COMMENT
 )
VALUES
(
  :F_N_NATIONKEY
 ,:F_N_NAME
 ,:F_N_REGIONKEY
 ,:F_N_COMMENT
);

END LOADING;

.LOGOFF;

---------------------------------------
inmod.region
---------------------------------------

SESSIONS 1;
REMOTE dino4 root /home2/tpcd/pfast/inmod/region/region;

LOGON local/tpcd100g,tpcd100g;

DROP TABLE REGION;
DROP TABLE ERRREGIONTMP1;
DROP TABLE ERRREGIONTMP2;

CREATE MULTISET TABLE REGION
  (
   R_REGIONKEY INTEGER NOT NULL
  ,R_NAME VARCHAR(25) CASESPECIFIC NOT NULL
  ,R_COMMENT VARCHAR(152) CASESPECIFIC NOT NULL
  )
UNIQUE PRIMARY INDEX ( R_REGIONKEY );

BEGIN LOADING REGION
      ERRORFILES ERRREGIONTMP1, ERRREGIONTMP2;

DEFINE
   F_R_REGIONKEY (INTEGER)
  ,F_R_NAME (CHAR(25))
  ,F_R_COMMENT (VARCHAR(152))
INMOD=BLKEXIT;

SHOW;

INSERT INTO REGION
 (
   R_REGIONKEY
  ,R_NAME
  ,R_COMMENT
 )
VALUES
(
  :F_R_REGIONKEY
 ,:F_R_NAME
 ,:F_R_COMMENT
);

END LOADING;

.LOGOFF;
```

Appendix G. Driver and Implementation Specific Layer Detail

Driver program

The driver program "tpcddriver" was written to simplify running the benchmark. It incorporates all of the QGEN program, plus those parts of the DBGEN program related to generating update sets. This driver is capable of running the entire benchmark, including both the power test and the throughput tests. It uses standard Teradata call-level interface (CLIv2) to send requests to the Teradata system. Because CLIv2 operates at a very low level, an implementation specific layer was added between the driver and CLIv2 to hide some of the complexities of CLIv2 from the main body of the tpcddriver.
The file tpcddriver.c is the main program for the driver. Vsub.c is the same as the standard varsub.c from the standard QGEN code, with modifications to make the substitutions in a buffer instead of writing them to disk. The files build.c, bm_utils.c, rnd.c, and bcd2.c are used unmodified from the standard DBGEN distribution. The include files sqltypes.h and sql.h are standard include files to define X/Open CLI style of interface. These can be found from either the ISO CLI spec (ISO/IEC 9075-3:1995) or from the Microsoft ODBC SDK.

Makefile

```
#  Sccsid:     @(#)makefile.suite     9.1.1.13     12/14/95  16:06:25
# makefile for DSS benchamrk data generator
#
#
#################
## CHANGE NAME OF ANSI COMPILER HERE
#################
CC      = cc
# Current values for DATABASE are: INFORMIX, DB2, TDAT (Teradata),
#                                  SQLSERVER, SYBASE
# Current values for PLATFORM are: ATT, DOS, HP, IBM, ICL, MVS,
#                                  SGI, U2200, VMS, WIN32
DATABASE=TDAT
PLATFORM=ATT
#
CFLAGS  = -O -D$(DATABASE) -D$(PLATFORM) -DSTDLIB_HAS_GETOPT -D__STDC__
-o1 -g -586 -I/usr/include -I/usr/ucbinclude

LDFLAGS =
# The OBJ,EXE and LIB macros will need to be changed for compilation
under
#  Windows NT
OBJ     = .o
EXE     =
LIBS    = -lcliv2 -ltdusr -lfast -lsocket -lnsl  -lm -lc -lgen -lx
#
# NO CHANGES SHOULD BE NECESSARY BELOW THIS LINE
################

PROG1 = dbgen$(EXE)
PROG2 = qgen$(EXE)
PROG3 = tpcddriver

HDR1 = dss.h rnd.h config.h dsstypes.h shared.h bcd2.h
SRC1 = build.c driver.c bm_utils.c rnd.c print.c load_stub.c bcd2.c
speed_seed.c
OBJ1 = build$(OBJ) driver$(OBJ) bm_utils$(OBJ) rnd$(OBJ) print$(OBJ)
load_stub$(OBJ) bcd2$(OBJ) speed_seed$(OBJ)
SETS = dists.dss
DDL  = dss.ddl dss.ri
OTHER=makefile.suite $(SETS) $(DDL) 10001001
DOC=README History Porting.Notes FAQ
DBGENSRC=$(SRC1) $(HDR1) $(OTHER) $(DOC) $(SRC2) $(HDR2) $(SRC3)
$(HDR3)
SRC2 = qgen.c varsub.c
HDR2 = tpcd.h
OBJ2 = build$(OBJ) bm_utils$(OBJ) qgen$(OBJ) rnd$(OBJ) varsub$(OBJ)
HDR3 = dss.h rnd.h config.h dsstypes.h shared.h bcd2.h tdatsql.h
tsqlt.h
SRC3 = build.c tpcddriver.c bm_utils.c rnd.c vsub.c bcd2.c tdatsql.c
tsqlt.c tdatload.c
OBJ3 = build$(OBJ) tpcddriver$(OBJ) bm_utils$(OBJ) rnd$(OBJ) vsub$(OBJ)
bcd2$(OBJ) tdatsql$(OBJ) tsqlt$(OBJ) tdatload$(OBJ)


all: $(PROG1) $(PROG2) $(PROG3)

$(PROG1): $(OBJ1) $(SETS)
        $(CC) $(LDFLAGS) -o $@ $(OBJ1) $(LIBS)

$(PROG2): $(OBJ2)
        $(CC) $(LDFLAGS) -o $@ $(OBJ2) $(LIBS)

$(PROG3): $(OBJ3) $(SETS)
        $(CC) $(LDFLAGS) -o $@ $(OBJ3) $(LIBS)

clean:
        rm -f  $(PROG1) $(PROG2) $(PROG3) $(OBJ1) $(OBJ2) $(OBJ3)
```

```
lint:
	lint $(CFLAGS) $(LDFLAGS) $(SRC3)
tar: $(DBGENSRC)
	tar cvhf $(PROG1).tar $(DBGENSRC)
dbgenshar: $(DBGENSRC)
	shar -o dbgen.shar $(DBGENSRC)
dbgendisk: $(DBGENSRC)
	for f in $(DBGENSRC) ; \
	do  \
	unix2dos $$f /pcfs/$$f ; \
	done
portable:
	for f in $(DBGENSRC) ; \
	do  \
	expand $$f > /tmp/foo; \
	mv -f /tmp/foo $$f; \
	awk 'length > 72 { print FILENAME ":" NR " too long " }' $$f ;
\
	done
rnd$(OBJ): rnd.h
$(OBJ1): config.h makefile.comp
$(OBJ1): dss.h dsstypes.h
$(OBJ2): dss.h tpcd.h config.h
$(OBJ3): dss.h tpcd.h config.h dsstypes.h tdatsql.h
```

tpcddriver.c

```
/*
 * tpcddriver.c  This program is a master driver program that runs
 * the entire TPC-D benchmark.  It first runs the power tests, then
 * it runs the throughput tests.  It includes the queries and the
 * update functions UF1 and UF2.
 * This program incorporates all of QGEN.  It also has some of
 * DBGEN incorporated so it can generate update data on the fly.
 */
#define DECLARER
#define NO_FUNC (int (*) ()) NULL     /* to clean up tdefs */
#define NO_LFUNC (long (*) ()) NULL          /* to clean up tdefs */
#define LIFENOISE(cnt) if (i % cnt == 0 && (flags & VERBOSE))
fprintf(stderr, ".")
#define MAXSESS 500
#define EVERYTHING        0x10000

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <limits.h>
#include <math.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include <signal.h>
#include <errno.h>
#if (defined(_POSIX_)||!defined(WIN32))
#include <unistd.h>
#include <sys/wait.h>
#define INLINE _Inline
#else
#include <process.h>
#define INLINE _inline
#ifndef timeb

#define timeb _timeb
#endif
#ifndef itoa
#define itoa _itoa
#endif
#ifndef ftime
#define ftime _ftime
#endif
#ifndef sleep
#define sleep(x) Sleep(x*1000)
typedef unsigned long DWORD;
__declspec(dllimport) void __stdcall Sleep(DWORD dwMillseconds);
#endif

#endif
#include <ctype.h>
#include <time.h>
#include <assert.h>

#undef FAR
#define FAR

typedef void *HWND;
#include <sqltypes.h>
#include <sql.h>
#include <sqlext.h>
#include <coptypes.h>
#include <dbcarea.h>
#include <parcel.h>
#undef VERSION

/* includes from QGEN */
#include "config.h"
#include "dss.h"
#include "tpcd.h"
/* These two from DBGEN */
#include "dsstypes.h"
#include "bcd2.h"


SQLRETURN SQL_API SQLSetUsingParcel (

	SQLHSTMT hstmt,

	char *uptr,

	long ulen);
```

```
        SQLRETURN SQL_API SQLWaitForAny (
                                SQLHENV henv,

        SQLHSTMT * lphstmt,
                                                                int
*stmtno);

#ifndef BOOL
typedef int BOOL;
#endif
#ifndef TRUE
#define TRUE 1
#endif
#ifndef FALSE
#define FALSE 0
#endif
long throughputstreams;
int ac1;
char **av1;
int pids[MAX_CHILDREN];

SQLHENV henv;
SQLHDBC hdbcs[MAXSESS + 1];
SQLHSTMT hstmts[MAXSESS + 1];
BOOL hstmtbusy[MAXSESS + 1];
SQLHDBC hdbc;
SQLHSTMT hstmt;

#define LINE_SIZE 512

/*
 * Function Protoypes
 */

int prep_direct(char *);
int close_direct(void);
int ld_line(order_t * o, int mode);
int ld_order(order_t * o, int mode);
int ld_order_line(order_t * o, int mode);
int ld_deletes (long okey, int mode);

void vsub PROTO ((char *SQLRequest, int qnum, int vnum, int flags));
int strip_comments PROTO ((char *line));
void usage PROTO ((void));
int process_options PROTO ((int cnt, char **args));
int setup PROTO ((void));
void qsub PROTO ((char *qtag, int flags));
void gen_updates (long start, long count, long upd_num);
int gen_deletes (long min, long cnt, long num, int special);



extern char *optarg;
extern int optind;
char **mk_ascdate (void);
extern long Seed[];
```

```
char **asc_date;
int snum = -1;
int use_fastload_for_updates = 3;
char *prog;
char *p1 = NULL;
FILE *fdetail;
FILE *fsummary;


int xld_order (order_t * o, int mode);

tdef tdefs[] =
{
        {"part.tbl", "part table", 200000, NO_FUNC,
         {NO_FUNC, NO_FUNC}, NO_LFUNC, NONE},
        {"partsupp.tbl", "partsupplier table", 200000, NO_FUNC,
         {NO_FUNC, NO_FUNC}, NO_LFUNC, NONE},
        {"supplier.tbl", "suppliers table", 10000, NO_FUNC,
         {NO_FUNC, NO_FUNC}, NO_LFUNC, NONE},
        {"customer.tbl", "customers table", 150000, NO_FUNC,
         {NO_FUNC, NO_FUNC}, NO_LFUNC, NONE},
        {"order.tbl", "order table", 150000, NO_FUNC,
         {NO_FUNC, NO_FUNC}, NO_LFUNC, NONE},
        {"lineitem.tbl", "lineitem table", 150000, NO_FUNC,
         {NO_FUNC, NO_FUNC}, NO_LFUNC, NONE},
        {"order.tbl", "order/lineitem tables", 150000, NO_FUNC,
         {NO_FUNC, NO_FUNC}, NO_LFUNC, LINE},
        {"part.tbl", "part/partsupplier tables", 200000, NO_FUNC,
         {NO_FUNC, NO_FUNC}, NO_LFUNC, PSUPP},
        {"time.tbl", "time table", 2557, NO_FUNC,
         {NO_FUNC, NO_FUNC}, NO_LFUNC, NONE},
        {"nation.tbl", "nation table", NATIONS_MAX, NO_FUNC,
         {NO_FUNC, NO_FUNC}, NO_LFUNC, NONE},
        {"region.tbl", "region table", NATIONS_MAX, NO_FUNC,
         {NO_FUNC, NO_FUNC}, NO_LFUNC, NONE},
        {"TPCDSEED", NULL, 0, NO_FUNC,
         {NO_FUNC, NO_FUNC}, NO_LFUNC, NONE}
};

long SaveSeed[MAX_STREAM + 1];
long tempSeed;
long initSeeds[MAX_STREAM + 1];
long num_seeds = 0;
long rndm;
int rowcnt = 0;
long minrow = 0, upd_num = 0;
int setsPerReq = 1;
int maxSess = 56;
int specialtest = 0;
double flt_scale;
struct timeb StartTime;
double UF1time;
double QueryTime[QUERIES_PER_SET + 1];
double UF2time;
double Throughputtime;
```

```
char SQLRequest[32768];
char UsingBuffer[32768];


/* From DBGEN, no changes */
void
mk_sparse (long *low_res, long *high_res, long base, long seq)
{
        long low_mask, seq_mask, overflow = 0;
        int count = 0;

        low_mask = (1 << SPARSE_KEEP) - 1;
        seq_mask = (1 << SPARSE_BITS) - 1;
        bin_bcd2 (base, low_res, high_res);
        bcd2_div (low_res, high_res, 1 << SPARSE_KEEP);
        bcd2_mul (low_res, high_res, 1 << SPARSE_BITS);
        bcd2_add (low_res, high_res, seq);
        bcd2_mul (low_res, high_res, 1 << SPARSE_KEEP);
        bcd2_add (low_res, high_res, base & low_mask);
        bcd2_bin (&low_mask, *low_res);
        bcd2_bin (&seq_mask, *high_res);
        *low_res = low_mask;
        *high_res = seq_mask;
        return;
}



/*
 * FUNCTION strip_comments(line)
 *
 * remove all comments from 'line'; recognizes both {} and -- comments
 */
int
strip_comments (char *line)
{
        static int in_comment = 0;
        char *cp1, *cp2;

        cp1 = line;

#ifdef WIN32
/* Disable warning about conditional expression is constant */
#pragma warning(disable:4127)
#endif
        while (1)                                       /* traverse the
entire string */
                {
#ifdef WIN32
#pragma warning(default:4127)
#endif
                        if (in_comment)
                                {
                                        if ((cp2 = strchr (cp1, '}')) !=
NULL)   /* comment ends */
                                                {
                                                        strcpy (cp1, cp2 +
1);
                                                        in_comment = 0;
                                                        continue;
                                                }
                                        else
                                                {
                                                        *cp1 = '\0';
                                                        break;
                                                }
                                }
                        else
                                /* not in_comment */
                                {
                                        if ((cp2 = strchr (cp1, '-')) !=
NULL)
                                                {
                                                        if (*(cp2 + 1) ==
'-')    /* found a '--' comment */
                                                                {

        *cp2 = '\0';

        break;
                                                                }
                                                }
                                        if ((cp2 = strchr (cp1, '{')) !=
NULL)   /* comment starts */
                                                {
                                                        in_comment = 1;
                                                        *cp2 = ' ';
                                                        continue;
                                                }
                                        else
                                                break;
                                }
                }
        return (0);
}

/*
 * FUNCTION qsub(char *qtag, int flags)
 *
 * based on the settings of flags, and the template file $QDIR/qtag.sql
 * make the following substitutions to turn a query template into EQT
 *
 *  String      Converted to            Based on
 *  ======      ============            ===========
 *  first line  database <db_name>;      -n from command line
 *  second line set explain on;         -x from command line
 *   :<number>  parameter <number>
 *  :k          set number
 *  :o          output to outpath/qnum.snum
 *                                      -o from command line,
SET_OUTPUT
 *  :s          stream number
 *  :b          BEGIN WORK;             -a from command line,
START_TRAN
 *  :e          COMMIT WORK;            -a from command line, END_TRAN
 *  :q          query number
```

```
 *  :n<number>                          sets rowcount to be returned
 */
void
qsub (char *qtag, int flags)
{
        static char *line = NULL, *qpath = NULL;
        int qnum;
        int i;
        FILE *qfp;
        char *cptr, *mark, *qroot = NULL;
        char tempStr[128];

        qnum = atoi (qtag);
        if (line == NULL)
                {
                        line = malloc (BUFSIZ);
                        qpath = malloc (BUFSIZ);
                        if (line == NULL || qpath == NULL)
                                {
                                        fprintf (stderr, "Malloc failed
(qsub)\n");
                                        fflush (stderr);
                                        fclose (fdetail);
                                        fclose (fsummary);
                                        exit (2);
                                }
                }

        qroot = env_config (QDIR_TAG, QDIR_DFLT);
        sprintf (qpath, "%s/%s.sql", qroot, qtag);
        qfp = fopen (qpath, "r");
        if (qfp == NULL)
                {
                        fprintf (stderr, "open failed for query template
%s, errno=%d\n", qpath, errno);
                        fflush (stderr);
                        fclose (fdetail);
                        fclose (fsummary);
                        exit (1);
                }

        SQLRequest[0] = '\0';
        rowcnt = rowcnt_dflt[qnum];
        vsub (SQLRequest, qnum, 0, flags);    /* set the variables */

        /*//if (flags & DFLT_NUM)
           //    { strcat(SQLRequest, SET_ROWCOUNT); strcat(SQLRequest,
itoa(rowcnt,tempStr,10)); }
         */
        while (fgets (line, BUFSIZ, qfp) != NULL)
                {
                        if (line[strlen (line) - 1] == 0x0A)  /* strip LF
chars */
                                {
                                        line[strlen (line) - 1] = ' ';
                                        strcat (line, "\r");
                                }
                        if (line[0] == '.')    /* BTEQ command? */
                                strcpy (line, " \r");
                        else if (line[0] == ' ' && line[1] == '.')
                                strcpy (line, " \r");


                        if (!(flags & COMMENT))
                                strip_comments (line);

                        mark = line;
                        while ((cptr = strchr (mark, VTAG)) != NULL)
                                {
                                        *cptr = '\0';
                                        cptr++;
                                        strcat (SQLRequest, mark);
                                        switch (*cptr)
                                                {
                                                case 'b':
                                                case 'B':
                                                        if ((flags & ANSI)
== 0)
                                                                {

        strcat (SQLRequest, START_TRAN);

        strcat (SQLRequest, "\r");
                                                                }
                                                        cptr++;
                                                        break;
                                                case 'c':
                                                case 'C':
                                                        /*if (flags &
DBASE)
                                                           //    {
strcat(SQLRequest, SET_DBASE); strcat(SQLRequest, db_name);   }
                                                         */
                                                        cptr++;
                                                        break;
                                                case 'e':
                                                case 'E':
                                                        if ((flags & ANSI)
== 0)
                                                                {

        strcat (SQLRequest, END_TRAN);

        strcat (SQLRequest, "\r");
                                                                }
                                                        cptr++;
                                                        break;
                                                case 'n':
                                                case 'N':
                                                        if (!(flags &
DFLT_NUM))
                                                                {

        rowcnt = atoi (++cptr);
```

```
        while (isdigit (*cptr) || *cptr == ' ')

        cptr++;

        /*sprintf(tempStr, SET_ROWCOUNT, rowcnt);

//strcat(SQLRequest, tempStr); */
                                                        }
                                                continue;
                                        case 'o':
                                        case 'O':
                                                if (flags &
OUTPUT)
                                                        {
        /*fprintf(ofp,"%s '%s/%s.%d'", SET_OUTPUT, osuff,

qtag, (snum < 0)?0:snum);  */
                                                        }
                                                cptr++;
                                                break;
                                        case 'q':
                                        case 'Q':
                                                strcat
(SQLRequest, qtag);
                                                cptr++;
                                                break;
                                        case 's':
                                        case 'S':
                                                sprintf (tempStr,
"%d", (snum < 0) ? 0 : snum);
                                                strcat
(SQLRequest, tempStr);
                                                cptr++;
                                                break;
                                        case 'X':
                                        case 'x':
                                                if (flags &
EXPLAIN)
                                                        {

        strcat (SQLRequest, GEN_QUERY_PLAN);

        strcat (SQLRequest, "\r");
                                                        }
                                                cptr++;
                                                break;
                                        default:
                                                vsub (SQLRequest,
qnum, atoi (cptr), flags & DFLT);
                                                while (isdigit
(*++cptr));
                                                break;
                                        }
                                mark = cptr;
                        }
                strcat (SQLRequest, mark);
        }
        fclose (qfp);
        for (i = strlen (SQLRequest) - 1; i > 0; i--)
                if (SQLRequest[i] == ' ' ||
                        SQLRequest[i] == '\r' ||
                        SQLRequest[i] == '\t')
                        SQLRequest[i] = '\0';
                else
                        break;
        if (SQLRequest[strlen (SQLRequest) - 1] == ';')
                SQLRequest[strlen (SQLRequest) - 1] = '\0';
        return;

}

void
usage (void)
{
        printf("TPC-D Benchmark Driver, matched to DBGEN/QGEN (Version
%d.%d.%d%c)\n",
                        VERSION,RELEASE,
                            MODIFICATION,PATCH);

        printf ("Portions Copyright %s %s\n", TPC, C_DATES);
        printf ("USAGE: %s <options> [ queries ]\n", prog);
        printf ("Options:\n");
        printf ("\t-c\t\t-- retain comments found in template.\n");
        printf ("\t-d\t\t-- use default substitution values.\n");
        printf ("\t-e\t\t-- run everything: Power test and throughput
test.\n");
        printf ("\t-h\t\t-- print this usage summary.\n");
/*printf("\t-i <str>\t-- use file <str> for initialization.\n");
   //printf("\t-l <str>\t-- log parameters to <str>.\n"); */
        printf ("\t-m <n>\t\t-- number of sessions to use for
updates.\n");
        printf ("\t-M <n>\t\t-- number of transactions per multi-
statement request for updates.\n");
        printf ("\t-n <str>\t-- connect to Teradata using logon
<str>.\n");
        printf ("\t-N\t\t-- use default rowcounts and ignore :n
directive.\n");
        printf ("\t-o <str>\t-- set the output file base path to
<str>.\n");
        printf ("\t-p <n>\t\t-- use the query permutation for stream
<n>\n");
        printf ("\t-r <n,n,>\t\t-- seed the random number generator
with <n>\n");
        printf ("\t-R  \t\t-- seed the random number generator with the
current time of day\n");
        printf ("\t-s <n>\t\t-- base substitutions and updates on an SF
of <n>\n");
        printf ("\t-S <n>\t\t-- set number of streams for throughput
test to <n>\n");
        printf ("\t-U <s>\t\t-- generate <s> update sets.  Use 0 to
avoid updates\n");
        printf ("\t-C <n>\t\t-- use seed file generated for <n> load
processes to generate updates\n");
        printf ("\t-v\t\t-- verbose.\n");
```

```
        printf("\t-T\t\t-- use TIME table values for date
substitutions\n");
        printf ("\t-x\t\t-- enable EXPLAIN in each query.\n");
}

int
process_options (int cnt, char **args)
{
        int flag;
        int i;

        while ((flag = getopt (cnt, args,
"acC:dehi:M:m:n:Nl:o:p:P:r:s:S:t:TU:vx")) != -1)
                switch (flag)
                        {
                        case 'a':                       /* special test
mode */
                                specialtest = 1;
                                flags |= VERBOSE;
                                break;
                        case 'c':                       /* retain comments
in EQT */
                                flags |= COMMENT;
                                break;
                        case 'C':
                                children = atoi (optarg);
                                if (children > MAX_CHILDREN)
                                        {
                                                fprintf (stderr, "Children
reset to limit of %d\n",

MAX_CHILDREN);
                                                children = MAX_CHILDREN;
                                        }
                                break;
                        case 'd':                       /* use default
substitution values */
                                flags |= DFLT;
                                break;
                        case 'e':                       /* Run everything,
power test + multisession test. */
                                snum = 0;
                                flags |= EVERYTHING;
                                break;

                        case 'h':                       /* just generate
the usage summary */
                                usage ();
                                exit (0);
                                break;
                        case 'i':                       /* set stream
initialization file name */
                                ifile = malloc (strlen (optarg) + 1);
                                MALLOC_CHECK (ifile);
                                strcpy (ifile, optarg);
                                flags |= INIT;
                                break;
                        case 'l':                       /* log parameter
usages */
                                lfile = malloc (strlen (optarg) + 1);
                                MALLOC_CHECK (lfile);
                                strcpy (lfile, optarg);
                                flags |= LOG;
                                break;
                        case 'm':                       /* sessions to use
for updates. */
                                maxSess = atoi (optarg);
                                if (maxSess < 1 || maxSess > MAXSESS)
                                        {
                                                fprintf (stderr, " Max
sessions out of range, ignored\n");
                                                maxSess = 56;
                                        }
                                break;

                        case 'M':                       /* transactions
per multi-statement req. */
                                setsPerReq = atoi (optarg);
                                if (setsPerReq < 1 || setsPerReq > 9)
                                {
                                        fprintf (stderr, " Sets per
Request out of range, ignored\n");
                                        setsPerReq = 1;
                                }
                                break;

                        case 'N':                       /* use default
rowcounts */
                                flags |= DFLT_NUM;
                                break;
                        case 'n':                       /* set database
name */
                                db_name = malloc (strlen (optarg) + 1);
                                MALLOC_CHECK (db_name);
                                strcpy (db_name, optarg);
                                flags |= DBASE;
                                break;
                        case 'o':                       /* set the output
path */
                                osuff = malloc (strlen (optarg) + 1);
                                MALLOC_CHECK (osuff);
                                strcpy (osuff, optarg);
                                flags |= OUTPUT;
                                break;
                        case 'p':                       /* permutation for
a given stream */
                                snum = atoi (optarg);
                                break;
                        case 'P':
                                use_fastload_for_updates = atoi (optarg);
                                break;
                        case 'r':                       /* set random
number seed for parameter gen */
                                flags |= SEED;
```

```
                                num_seeds = sscanf (optarg,
"%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld"

                "%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld"

                "%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld"

                "%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld",

        &initSeeds[0], &initSeeds[1], &initSeeds[2], &initSeeds[3],
&initSeeds[4], &initSeeds[5],

        &initSeeds[6], &initSeeds[7], &initSeeds[8], &initSeeds[9],
&initSeeds[10], &initSeeds[11],

        &initSeeds[12], &initSeeds[13], &initSeeds[14], &initSeeds[15],
&initSeeds[16], &initSeeds[17],

        &initSeeds[18], &initSeeds[19], &initSeeds[20], &initSeeds[21],
&initSeeds[22], &initSeeds[23],

        &initSeeds[24], &initSeeds[25], &initSeeds[26], &initSeeds[27],
&initSeeds[28], &initSeeds[29],

        &initSeeds[30], &initSeeds[31], &initSeeds[32], &initSeeds[33],
&initSeeds[34], &initSeeds[35],

        &initSeeds[36], &initSeeds[37], &initSeeds[38], &initSeeds[39],
&initSeeds[40], &initSeeds[41],

        &initSeeds[42], &initSeeds[43], &initSeeds[44], &initSeeds[45],
&initSeeds[46], &initSeeds[47]);
                                rndm = initSeeds[0];
                                if (num_seeds > 1)
                                        {
                                                for (i = 0; i < num_seeds;
i++)
                                                        {
                                                                //printf ("
-- Using %ld as seed for stream %d\n", initSeeds[i], i);
                                                                if
(initSeeds[i] == 0)
                                                                        {

        fprintf (stderr, " -- Seeds cannot be zero!\n");

        exit (1);
                                                                        }
                                                        }
                                        }
                                //else
                                //      printf (" -- Using %ld as
seed\n", rndm);
                                break;

                        case 's':                       /* scale of data
set to run against */
                                flt_scale = (double) atof (optarg);
                                if (flt_scale < 1.0)
                                        {
                                                int i;

                                                scale = 1;
                                                for (i = PART; i < TIME;
i++)
                                                        {

        tdefs[i].base = (long) (tdefs[i].base * flt_scale);
                                                                if
(tdefs[i].base < 1)

        tdefs[i].base = 1;
                                                        }
                                        }
                                else
                                        scale = (long) flt_scale;
                                if (scale < 1)
                                        {
                                                fprintf (stderr,
                                                 "WARNING: Scale
set to its lower bound (1)\n");
                                                scale = 1;
                                        }
                                break;

                        case 'S':
                                throughputstreams = atoi (optarg);
                                if (throughputstreams >= MAX_STREAM)
                                        {
                                                fprintf (stderr,
"Throughput streams reset to limit of %d\n",

MAX_STREAM);
                                                throughputstreams =
MAX_STREAM;
                                        }
                                break;
                        case 't':                       /* set termination
file name */
                                tfile = malloc (strlen (optarg) + 1);
                                MALLOC_CHECK (tfile);
                                strcpy (tfile, optarg);
                                flags |= TERMINATE;
                                break;
                        case 'T':                       /* use time table
*/
                                oldtime = 1;
                                break;
                        case 'U':                       /* generate for
update stream */
                                updates = atoi (optarg);
                                break;
                        case 'v':                       /* verbose */
                                flags |= VERBOSE;
                                break;
```

```
                        case 'x':                       /* set explain in
the queries */
                                flags |= EXPLAIN;
                                break;
                        default:
                                printf ("unknown option '%s' ignored\n",
args[optind]);
                                usage ();
                                exit (1);
                                break;
                        }
        return (0);
}

int
setup (void)
{
        asc_date = mk_ascdate ();
        read_dist (env_config (DIST_TAG, DIST_DFLT), "p_cntr",
&p_cntr_set);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "colors",
&colors);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "p_types",
&p_types_set);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "nations",
&nations);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "nations2",
&nations2);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "regions",
&regions);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "o_oprio",
                        &o_priority_set);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "instruct",
                        &l_instruct_set);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "smode",
&l_smode_set);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "category",
                        &l_category_set);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "rflag",
&l_rflag_set);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "msegmnt",
&c_mseg_set);

        return (0);
}

BOOL canRetry(long ErrorCode)
{
        if (ErrorCode == 2825 || /* No record of transaction after DBC
restart */
                ErrorCode == 2826 || /* Request complete, but output
lost due to DBC Restart */
                ErrorCode == 2584 || /* Transaction aborted due to Disk
I/O error */
                ErrorCode == 2631 || /* Transaction aborted due to
deadlock */
                ErrorCode == 3120 || /* Request lost due to DBC recovery
*/
                ErrorCode == 3679 || /* Transaction aborted due to
resource shortage */
                ErrorCode == 3897       /* Transaction aborted due to
DBC restart */ )
                return TRUE;
        else
                return FALSE;
}

/* First, some helper routines to copy data into the parcel we are
building */
static void
DR_Strt ()
{
        p1 = UsingBuffer;                       /* Skip 2 for the length
field */

}
static void
DR_End (int t)
{
        if (t);
        if ((p1 - UsingBuffer) > 20000)
                fprintf (stderr, " Using data too long!!!!!!!\n");
        if ((p1 - UsingBuffer) + strlen (SQLRequest) > 32000)
                fprintf (stderr, " Using data too long!!!!!!!\n");
        SQLSetUsingParcel (hstmt, UsingBuffer, (int) (p1 -
UsingBuffer));
}

INLINE
static void
DR_Int (long x)
{
        memcpy (p1, &x, 4);
        p1 += 4;
}
INLINE
static void
DR_Int64 (long xlow, long xhigh)      /* Used only for orderkey, which
gets > 32 bits at times */
{
        double tempx;
        unsigned long templ;
        long temph;
        if (xhigh == 0)
                {
                        memcpy (p1, &xlow, 4);
                        p1 += 4;
                        memcpy (p1, &xhigh, 4);
                        p1 += 4;
                }
        else
                {
                        /* Sigh, don't have an Int64 type, so need to
use double and convert to 2 longs */
                        tempx = xhigh * 10000000.0 + xlow;
                        temph = (long) (tempx / 4294967296.0);
```

```
                        templ = (unsigned long) (tempx - temph *
4294967296.0);
                        memcpy (p1, &templ, 4);
                        p1 += 4;
                        memcpy (p1, &temph, 4);
                        p1 += 4;

                }

}
INLINE
static void
DR_VStr (char *x)
{
        short len;
        len = (short) strlen (x);
        memcpy (p1, &len, 2);
        p1 += 2;
        memcpy (p1, x, len);
        p1 += len;
}
INLINE
static void
DR_Str (char *x, size_t y)
{
        memset (p1, ' ', y);
        memcpy (p1, x, strlen (x));
        p1 += y;
}
INLINE
static void
DR_Money (long x)
{
        long temp = 0;
        memcpy (p1, &x, 4);
        p1 += 4;
        memcpy (p1, &temp, 4);
        p1 += 4;

}
INLINE
static void
DR_Char (char x)
{
        *p1 = x;
        p1++;
}
INLINE
static void
DR_Date (char *x)
{
        long tempdate;
        int y, m, d;
        y = atoi (x);
        m = atoi (x + 5);
        d = atoi (x + 8);
        if (y < 1800 || y > 2200 || m < 1 || m > 12 || d < 1 || d > 31)
                {
                        fprintf (stderr, " Help! I can't interpret the
date %s\n", x);
                        exit (1);
                }
        tempdate = (y - 1900) * 10000 + m * 100 + d;
        memcpy (p1, &tempdate, 4);
        p1 += 4;

}

/*
   CREATE TABLE TPCD.ORDER  ( O_ORDERKEY       INTEGER NOT NULL,
   O_CUSTKEY        INTEGER NOT NULL,
   O_ORDERSTATUS    CHAR(1) NOT NULL,
   O_TOTALPRICE     DECIMAL(15,2) NOT NULL,
   O_ORDERDATE      DATE NOT NULL,
   O_ORDERPRIORITY  CHAR(15) NOT NULL,  -- R
   O_CLERK          CHAR(15) NOT NULL,  -- R
   O_SHIPPRIORITY   INTEGER NOT NULL,
   O_COMMENT        VARCHAR(79) NOT NULL);
 */

int
xld_order (order_t * p, int mode)
{
        if (mode);

        if (scale > 300)
                {
                        DR_Int64 (p->okey_low, p->okey_high);
                }
        else
                {
                        if (p->okey_high == 0)
                                DR_Int (p->okey_low);
                        else
                                DR_Int (p->okey_high * 10000000 + p-
>okey_low);
                }
        DR_Int (p->custkey);
        DR_Char (p->orderstatus);
        DR_Money (p->totalprice);
        DR_Date (p->odate);
        DR_Str (p->opriority, O_OPRIO_LEN);
        DR_Str (p->clerk, O_CLRK_LEN);
        DR_Int (p->spriority);
        DR_VStr (p->comment);
        DR_End (ORDER);
        return (0);
}

/*
   CREATE TABLE TPCD.LINEITEM ( L_ORDERKEY    INTEGER NOT NULL,
   L_PARTKEY     INTEGER NOT NULL,
   L_SUPPKEY     INTEGER NOT NULL,
   L_LINENUMBER  INTEGER NOT NULL,
   L_QUANTITY    DECIMAL(15,2) NOT NULL,
   L_EXTENDEDPRICE  DECIMAL(15,2) NOT NULL,
```

```
   L_DISCOUNT    DECIMAL(15,2) NOT NULL,
   L_TAX         DECIMAL(15,2) NOT NULL,
   L_RETURNFLAG  CHAR(1) NOT NULL,
   L_LINESTATUS  CHAR(1) NOT NULL,
   L_SHIPDATE    DATE NOT NULL,
   L_COMMITDATE  DATE NOT NULL,
   L_RECEIPTDATE DATE NOT NULL,
   L_SHIPINSTRUCT CHAR(25) NOT NULL,  -- R
   L_SHIPMODE     CHAR(10) NOT NULL,  -- R
   L_COMMENT      VARCHAR(44) NOT NULL);
 */
xld_one_line (order_t * p, int i, int mode)
{
        //int i;
        if (mode);
        //for (i = 0; i < p->lines; i++)
        //      {

                        if (scale > 300)
                                {
                                        DR_Int64 (p->l[i].okey_low, p-
>l[i].okey_high);
                                }
                        else
                                {
                                        if (p->l[i].okey_high == 0)
                                                DR_Int (p->l[i].okey_low);
                                        else
                                                DR_Int (p->l[i].okey_high
* 10000000 + p->l[i].okey_low);
                                }
                        DR_Int (p->l[i].partkey);
                        DR_Int (p->l[i].suppkey);
                        DR_Int (p->l[i].lcnt);
                        DR_Int (p->l[i].quantity);
                        DR_Money (p->l[i].eprice);
                        DR_Money (p->l[i].discount);
                        DR_Money (p->l[i].tax);
                        DR_Char (p->l[i].rflag[0]);
                        DR_Char (p->l[i].lstatus[0]);
                        DR_Date (p->l[i].sdate);
                        DR_Date (p->l[i].cdate);
                        DR_Date (p->l[i].rdate);
                        DR_Str (p->l[i].shipinstruct, L_INST_LEN);
                        DR_Str (p->l[i].shipmode, L_SMODE_LEN);
                        DR_VStr (p->l[i].comment);
                        DR_End (LINE);
        //      }
        return (0);
}

/*
int
xld_line (order_t * p,  int mode)
{
        int i;
        xld_order (p, mode);
        for (i = 0; i < p->lines; i++)
                xld_one_line (p, i, mode);
        return (0);
}

int
xld_order_line (order_t * p, int mode)
{
        xld_order (p, mode);
        xld_line (p,  mode);

        return (0);
} */


int
Fetch_and_Display_All_Rows (SQLHSTMT hstmt, int qnum)
{
        int rc;
        int rc2;
        char dValue[512];
        char ColTitle[60];
        short ColTitleLen;
        long actualLen;
        short icol;
        short maxcol;
        long totalRows;
        long columnDisplaySize[200];
        long columnSQLType[200];
        long currow;
        long temp;
        char SqlState[6];
        char ErrMsg[256];
        short ErrMsgLen;
        long ErrCode;


        if (SQLRowCount (hstmt, &totalRows) != SQL_SUCCESS)
              fprintf (stderr, " SQLRowCount problem\n");
        if (SQLNumResultCols (hstmt, &maxcol) != SQL_SUCCESS)
              fprintf (stderr, " SQLNumResultCols problem\n");
        if (maxcol == 0)
                {
                        fprintf (fdetail, "\n SQL statement
complete.\n");
                        fprintf (stdout, " SQL statement complete.\n");
                }
        else
                {
                        if (qnum > 0)
                                {
                                        fprintf (fdetail, "\n Query %d
complete, %d rows returned\n", qnum, totalRows);
                                        fprintf (stdout, " Query %d
complete, %d rows returned\n", qnum, totalRows);
                                }
                        else
                                {
```

```
                                        fprintf (fdetail, "\n Query
complete, %d rows returned\n", totalRows);
                                        fprintf (stdout, " Query
complete, %d rows returned\n", totalRows);
                                }

                        if (!(flags & EXPLAIN))
                                if (rowcnt > 0 && rowcnt < totalRows)
                                        {
                                                fprintf (fdetail, " only
displaying %d rows per TPC-D spec\n", rowcnt);
                                                totalRows = rowcnt;
                                        }
                        fprintf (fdetail, "\n\n");
                }
        currow = 0;
        actualLen = 0;
        if (totalRows != 0 && maxcol > 0)
                {
                        for (icol = 1; icol <= maxcol; icol++)
                                {
                                        ColTitle[0] = '\0';
                                        if (SQLColAttributes (hstmt,
icol, SQL_COLUMN_LABEL, ColTitle, 60, &ColTitleLen, &temp) !=
SQL_SUCCESS)
                                                {
                                                        SQLError
((SQLHENV) 0, (SQLHDBC) 0, hstmt, (SQLCHAR *) SqlState, &ErrCode,
(SQLCHAR *) ErrMsg, 256, &ErrMsgLen);

                                                        fprintf (stderr, "
ColAttributes call failed\n");
                                                        fprintf (stderr, "
%s (%ld) %s\n", SqlState, ErrCode, ErrMsg);
                                                }
                                        if (SQLColAttributes (hstmt,
icol, SQL_COLUMN_DISPLAY_SIZE, NULL, 0, NULL, &columnDisplaySize[icol])
!= SQL_SUCCESS)
                                                {
                                                        SQLError
((SQLHENV) 0, (SQLHDBC) 0, hstmt, (SQLCHAR *) SqlState, &ErrCode,
(SQLCHAR *) ErrMsg, 256, &ErrMsgLen);

                                                        fprintf (stderr, "
ColAttributes call failed\n");
                                                        fprintf (stderr, "
%s (%ld) %s\n", SqlState, ErrCode, ErrMsg);
                                                }
                                        if (SQLColAttributes (hstmt,
icol, SQL_COLUMN_TYPE, NULL, 0, NULL, &columnSQLType[icol]) !=
SQL_SUCCESS)
                                                {
                                                        SQLError
((SQLHENV) 0, (SQLHDBC) 0, hstmt, (SQLCHAR *) SqlState, &ErrCode,
(SQLCHAR *) ErrMsg, 256, &ErrMsgLen);

                                                        fprintf (stderr, "
ColAttributes call failed\n");
                                                        fprintf (stderr, "
%s (%ld) %s\n", SqlState, ErrCode, ErrMsg);
                                                }
                                        fprintf (fdetail, "%c", '|');
                                        actualLen = strlen (ColTitle);
                                        if (columnDisplaySize[icol] <
actualLen)
                                                columnDisplaySize[icol] =
actualLen;


                                        if (columnSQLType[icol] !=
SQL_CHAR && columnSQLType[icol] != SQL_VARCHAR)
                                                while
(columnDisplaySize[icol] > actualLen++ && actualLen < 81)
                                                        fprintf (fdetail,
"%c", '_');

                                        fprintf (fdetail, "%s",
ColTitle);

                                        if (columnSQLType[icol] ==
SQL_CHAR || columnSQLType[icol] == SQL_VARCHAR)
                                                while
(columnDisplaySize[icol] > actualLen++ && actualLen < 81)
                                                        fprintf (fdetail,
"%c", '_');
                                }
                        fprintf (fdetail, "|\n");
                }

        rc = SQL_SUCCESS;
        if (totalRows > 0)
                while (((rc = SQLFetch (hstmt)) == SQL_SUCCESS) &&
currow++ < totalRows)  /* records */
                        {
                                icol = 0;
                                actualLen = 0;
                                rc2 = SQL_SUCCESS;
                                while (((int) icol++ <= (int) maxcol) &&
                                        (rc2 = SQLGetData (hstmt,
icol, SQL_C_CHAR, dValue, 512, &actualLen) == SQL_SUCCESS))
                                        {
                                                fprintf (fdetail, "%c",
'|');
                                                if (columnSQLType[icol] !=
SQL_CHAR && columnSQLType[icol] != SQL_VARCHAR)
                                                        while
(columnDisplaySize[icol] > actualLen++ && actualLen < 81)
                                                                fprintf
(fdetail, "%c", ' ');

                                                fprintf (fdetail, "%s",
dValue);

                                                if (columnSQLType[icol] ==
SQL_CHAR || columnSQLType[icol] == SQL_VARCHAR)
```

```
                                                        while
(columnDisplaySize[icol] > actualLen++ && actualLen < 81)
                                                                fprintf
(fdetail, "%c", ' ');

                                        }
                                fprintf (fdetail, "|\n");
                                if (rc2 == SQL_ERROR)
                                        {
                                                SQLError ((SQLHENV) 0,
(SQLHDBC) 0, hstmt, (SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg,
256, &ErrMsgLen);

                                                fprintf (stderr, " GetData
call failed\n");
                                                fprintf (stderr, " %s
(%ld) %s\n", SqlState, ErrCode, ErrMsg);
                                        }
                        }
        fprintf (fdetail, "\n");
        if (rc == SQL_ERROR)
                {
                        SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmt,
(SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256, &ErrMsgLen);

                        fprintf (stderr, " Fetch call failed\n");
                        fprintf (stderr, " %s (%ld) %s\n", SqlState,
ErrCode, ErrMsg);
                }

        if (rc == SQL_ERROR)
                return ((SQLRETURN) rc);

        return (SQLFreeStmt (hstmt, SQL_CLOSE));
}

long start, last;

long orderRowsDeleted = 0;
long lineitemRowsDeleted = 0;
long orderRowsToBeDeleted = 0;
long orderRowsInError = 0;
int OrdersInReq[MAXSESS + 1] = {0};

typedef struct DBC
        {

                Int32 SessionId;
                Int32 ReqId;
                short numResultCols;
                long RowCount;
                char *p;
                unsigned short Async;
                unsigned short busy;
                long NativeError;
                char ErrorMessage[512];
                struct CliDataInfoType DataInfo;
                struct DBCAREA dbcarea;
        }
DBC;

typedef struct STMT
        {

                DBC *lpdbc;
                long hstmt_num;
        }
STMT;



SQLRETURN
Get_Delete_Answer (int h, SQLRETURN rc)
{
        long totalRows;
        char SqlState[6];
        char ErrMsg[256];
        short ErrMsgLen;
        long ErrCode;
        long j;


        if (rc == SQL_ERROR)
                {
                        SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmts[h],
(SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256, &ErrMsgLen);

                        if (canRetry(ErrCode)) /* Transaction ABORTed due
to DeadLock */
                                {
                                        fprintf (fdetail, " Deadlock
during delete\n");
                                        fprintf (fdetail, " %s (%ld)
%s\n", SqlState, ErrCode, ErrMsg);
                                        fprintf (stdout, " Deadlock
during delete\n");
                                        fprintf (stdout, " %s (%ld)
%s\n", SqlState, ErrCode, ErrMsg);
                                        SQLFreeStmt (hstmts[h],
SQL_CLOSE);
                                        fprintf (fdetail, " Attempting to
resubmit\n");
                                        fprintf (stdout, " Attempting to
resubmit\n");
                                        rc = SQLExecDirect(hstmts[h],
                                                (SQLCHAR FAR *)((STMT *)
hstmt)->lpdbc->dbcarea.req_ptr,
                                                (SQLINTEGER)   ((STMT *)
hstmt)->lpdbc->dbcarea.req_len);
                                        if (rc == SQL_ERROR)
                                        {
                                                fprintf (fdetail, "
Resubmit failed to work\n");
                                                fprintf (stdout, "
Resubmit failed to work\n");
```

```
                                        }
                                        else
                                                return
Get_Delete_Answer(h,rc);
                                }

                        orderRowsInError += OrdersInReq[h];
                        hstmtbusy[h] = FALSE;
                        DR_Strt ();

                        fprintf (fdetail, " Delete Request failed\n");
                        fprintf (fdetail, "%s\n", SQLRequest);
                        fprintf (fdetail, " %s (%ld) %s\n", SqlState,
ErrCode, ErrMsg);
                        fflush (fdetail);
                        fprintf (stderr, " Delete Request failed\n");
                        fprintf (stderr, "%s\n", SQLRequest);
                        fprintf (stderr, " %s (%ld) %s\n", SqlState,
ErrCode, ErrMsg);
                        SQLFreeStmt (hstmts[h], SQL_CLOSE);
                        fprintf (stderr, " Delete request failed, Cannot
continue!\n");
                        DR_Strt ();
                        return (-1);


                }
        else
                {
                        DR_Strt ();
                        if (flags & EXPLAIN)
                                {
                                        hstmtbusy[h] = FALSE;
                                        SQLSetStmtOption (hstmts[h],
SQL_ASYNC_ENABLE, SQL_ASYNC_ENABLE_OFF);
                                        fprintf (fdetail, "%s\n\n",
SQLRequest);
                                        Fetch_and_Display_All_Rows
(hstmts[h], -1);
                                        return (-1);
                                }
                        else
                                {
                                        long templong;


                                        hstmtbusy[h] = FALSE;
                                        SQLSetStmtOption (hstmts[h],
SQL_ASYNC_ENABLE, SQL_ASYNC_ENABLE_OFF);

                                        for (j = 1; j <= OrdersInReq[h];
j++)
                                                {
                                                        SQLRowCount
(hstmts[h], &totalRows);

                                                        orderRowsDeleted
+= totalRows;
                                                        if (totalRows != 1
&& totalRows != 0)
                                                                fprintf
(fdetail, "  %d rows deleted from Order Table\n", totalRows);
                                                        if (j !=
OrdersInReq[h])

                                                                if ((rc =
SQLMoreResults (hstmts[h])) != SQL_SUCCESS)
                                                                        {

        fprintf (stderr, " Problem with SQLMoreResult in del \n");

        if (rc == SQL_NO_DATA_FOUND)

                fprintf (stderr, " No data found from SQLMoreResults in
Del\n");

        else

                {

                        SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmts[h],
(SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256, &ErrMsgLen);

                        fprintf (stderr, " %s (%ld) %s\n", SqlState,
ErrCode, ErrMsg);


                        fprintf (fdetail, " Something is wrong in the
delete response\n");

                        fprintf (stderr, " Something is wrong in the
delete response\n");

                }


        break;
                                                                        }
                                                }
                                        //fprintf(stdout,"  %d rows
deleted from Order Table\n",totalRows);
                                        totalRows = 0;
                                        while ((rc = SQLMoreResults
(hstmts[h])) == SQL_SUCCESS)
                                                {
                                                        templong = 0;
                                                        SQLRowCount
(hstmts[h], &templong);
                                                        if (templong < 0
|| templong > 10)
                                                                fprintf
(fdetail, "  %d rows deleted from LineItem table\n", templong);

                                                        totalRows +=
templong;
```

```
                                        }
                                if (rc == SQL_ERROR)

                                        {
                                                fprintf (stderr, "
Problem with SQLMoreResult del2\n");
                                                SQLError
((SQLHENV) 0, (SQLHDBC) 0, hstmts[h], (SQLCHAR *) SqlState, &ErrCode,
(SQLCHAR *) ErrMsg, 256, &ErrMsgLen);
                                                fprintf (stderr, "
%s (%ld) %s\n", SqlState, ErrCode, ErrMsg);
                                        }
                                lineitemRowsDeleted += totalRows;
                        }


                }

        return 0;

}

int LinesPerOrder[MAXSESS + 1][20];
long orderRowsAdded = 0;
long lineitemRowsAdded = 0;
long orderRowsToBeAdded = 0;
long lineitemRowsToBeAdded = 0;
long duplicates = 0;
order_t o[MAXSESS + 1][20];


SQLRETURN
Get_Insert_Answer (int h, SQLRETURN rc)
{

        long totalRows;
        char SqlState[6];
        char ErrMsg[256];
        short ErrMsgLen;
        long ErrCode;
        int j;


        if (rc == SQL_ERROR)
                {
                        hstmtbusy[h] = FALSE;
                        SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmts[h],
(SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256, &ErrMsgLen);
                        if (ErrCode == 2801)   /* Duplicate unique
primary index */
                                {
                                        DR_Strt ();
                                        duplicates += OrdersInReq[h];
                                        fprintf (fdetail, " Duplicate on
orderkey %d\n", o[h][0].okey_low);
                                        fprintf (fdetail, " %s (%ld)
%s\n", SqlState, ErrCode, ErrMsg);
                                                fprintf (stdout, " Duplicate on
orderkey %d\n", o[h][0].okey_low);
                                                fprintf (stdout, " %s (%ld)
%s\n", SqlState, ErrCode, ErrMsg);
                                                SQLFreeStmt (hstmts[h],
SQL_CLOSE);
                                }
                        else if (canRetry(ErrCode))   /* Transaction
ABORTed due to DeadLock */
                                {
                                                fprintf (fdetail, " Deadlock on
orderkey %d\n", o[h][0].okey_low);
                                                fprintf (fdetail, " %s (%ld)
%s\n", SqlState, ErrCode, ErrMsg);
                                                fprintf (stdout, " Deadlock on
orderkey %d\n", o[h][0].okey_low);
                                                fprintf (stdout, " %s (%ld)
%s\n", SqlState, ErrCode, ErrMsg);
                                                SQLFreeStmt (hstmts[h],
SQL_CLOSE);
                                                fprintf (fdetail, " Attempting to
resubmit\n");
                                                fprintf (stdout, " Attempting to
resubmit\n");

                                                rc = SQLExecDirect(hstmts[h],
                                                        (SQLCHAR FAR *)((STMT *)
hstmt)->lpdbc->dbcarea.req_ptr,
                                                        (SQLINTEGER)   ((STMT *)
hstmt)->lpdbc->dbcarea.req_len);
                                                if (rc == SQL_ERROR)
                                                {
                                                        fprintf (fdetail, "
Resubmit failed to work\n");
                                                        fprintf (stdout, "
Resubmit failed to work\n");
                                                }
                                                else
                                                        return


Get_Insert_Answer(h,rc);
                                }
                        else
                                {
                                                DR_Strt ();
                                                orderRowsInError +=
OrdersInReq[h];
                                                fprintf (fdetail, " Insert
Request failed\n");
                                                fprintf (fdetail, "%s\n",
SQLRequest);
                                                fprintf (fdetail, " %s (%ld)
%s\n", SqlState, ErrCode, ErrMsg);
                                                fflush (fdetail);
                                                fprintf (stderr, " Insert Request
failed\n");
                                                fprintf (stderr, "%s\n",
SQLRequest);
                                                fprintf (stderr, " %s (%ld)
%s\n", SqlState, ErrCode, ErrMsg);
```

```
                                        SQLFreeStmt (hstmts[h],
SQL_CLOSE);
                                        fprintf (stderr, " Insert request
failed, Cannot continue!\n");
                                        DR_Strt ();
                                        OrdersInReq[h] = 0;
                                        return (-1);
                                }
                        OrdersInReq[h] = 0;
                }
        else
                {
                        DR_Strt ();
                        if (flags & EXPLAIN)
                                {
                                        hstmtbusy[h] = FALSE;
                                        SQLSetStmtOption (hstmts[h],
SQL_ASYNC_ENABLE, SQL_ASYNC_ENABLE_OFF);
                                        OrdersInReq[h] = 0;
                                        fprintf (fdetail, "%s\n\n",
SQLRequest);
                                        Fetch_and_Display_All_Rows
(hstmts[h], -1);
                                        return (-1);
                                }
                        else
                                {
                                        long templong;
                                        int k;

                                        hstmtbusy[h] = FALSE;
                                        SQLSetStmtOption (hstmts[h],
SQL_ASYNC_ENABLE, SQL_ASYNC_ENABLE_OFF);

                                        for (j = 0; j < OrdersInReq[h];
j++)
                                                {
                                                        SQLRowCount
(hstmts[h], &totalRows);
                                                        orderRowsAdded +=
totalRows;

                                                        if (totalRows !=
1)
                                                                fprintf
(fdetail, "  %d rows inserted into Order Table in this request! \n",
totalRows);
                                                        totalRows = 0;
                                                        for (k = 0; k <
LinesPerOrder[h][j]; k++)
                                                                {
                                                                        if
(SQLMoreResults (hstmts[h]) != SQL_SUCCESS)
        {

                fprintf (stderr, " Problem with SQLMoreResult for insert
\n");
                                SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmts[h], (SQLCHAR
*) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256, &ErrMsgLen);

                                fprintf (stderr, " %s (%ld) %s\n", SqlState, ErrCode,
ErrMsg);

        }

        templong = 0;
                                                                                if
(SQLRowCount (hstmts[h], &templong) != SQL_SUCCESS)

        fprintf (stderr, " Problem with SQLRowCount\n");
                                                                                if
(templong != 1)

        fprintf (fdetail, "  %d rows inserted into Lineitem Table in
this insert! \n", totalRows);



        totalRows += templong;
                                                                        }

                                                                lineitemRowsAdded
+= totalRows;
                                                                if (totalRows !=
LinesPerOrder[h][j])
                                                                        fprintf
(fdetail, "  %d rows inserted into LineItem table in this request,
should have been %d\n", totalRows, LinesPerOrder[h][j]);

                                                                rc =
SQLMoreResults (hstmts[h]);
                                                                if (rc ==
SQL_ERROR || rc == SQL_SUCCESS_WITH_INFO)
                                                                        {

        fprintf (stderr, " Problem with SQLMoreResult ins2\n");

        SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmts[h], (SQLCHAR *)
SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256, &ErrMsgLen);

        fprintf (stderr, " %s (%ld) %s\n", SqlState, ErrCode, ErrMsg);
                                                                        }
                                                                else if (rc ==
SQL_NO_DATA_FOUND && j < (OrdersInReq[h] - 1))
                                                                        {

        fprintf (stderr, " Problem with SQLMoreResults ins3\n");
                                                                        }
                                                                }
                                                        OrdersInReq[h] = 0;

                                                }
                                        }
        return 0;
```

```
}


void
Wait_For_Free_Hstmt (int *h, int inserts)
{
        int i;
        SQLRETURN rc;
        SQLHSTMT newhstmt;

        for (i = 1; i <= maxSess; i++)
                if (!hstmtbusy[i])
                        {
                                //fprintf(stderr,"Wait_for_Free returned
%d because it wasnt busy\n",i);
                                *h = i;
                                return;
                        }
        rc = SQLWaitForAny (henv, &newhstmt, &i);
        if (rc == SQL_ERROR && (i < 0 || newhstmt == SQL_NULL_HSTMT))
                {
                fprintf (stderr, "Unexpected error: SQLWaitForAny
returned SQL_ERROR in Wait_for_free\n");
                fprintf (fdetail, "Unexpected error: SQLWaitForAny
returned SQL_ERROR in Wait_for_free\n");

                return;

                }

        if (rc == SQL_NO_DATA_FOUND)
                {
                        fprintf (stderr, "Bug:  SQLWaitForAny returned
complete.\n");
                        *h = 1;
                        return;
                }
        if (hstmts[i] != newhstmt)
                fprintf (stderr, "SQLWaitForAny returned garbage\n");

        //fprintf(stderr,"SQLWaitForAny returned %d\n",i);
        if (inserts)

                Get_Insert_Answer (i, rc);
        else
                Get_Delete_Answer (i, rc);
        hstmtbusy[i] = FALSE;
        *h = i;
        return;
/*
   for (;;)
   {
   for (i=0;i<maxSess;i++)
   {
   if (Submit_Req(i)!=SQL_STILL_EXECUTING)
   {
   *h = i;
   return;
   }
   }
   }
 */
}

SQLRETURN
Submit_Req (int h, int inserts)
{

        SQLRETURN rc;



        if (!hstmtbusy[h])
                {
                        SQLSetStmtOption (hstmts[h], SQL_ASYNC_ENABLE,
SQL_ASYNC_ENABLE_ON);
                        hstmtbusy[h] = TRUE;
                }

        rc = SQLExecDirect (hstmts[h], (SQLCHAR *) SQLRequest,
SQL_NTS);
        if (rc == SQL_STILL_EXECUTING)
                {
                        //fprintf(stderr," Submit_req returns
SQL_STILL_EXECUTING on %d\n",h);
                        hstmtbusy[h] = TRUE;
                        return (rc);
                }
        //fprintf(stderr," Submit_req did not have to wait for
%d\n",h);

        SQLSetStmtOption (hstmts[h], SQL_ASYNC_ENABLE,
SQL_ASYNC_ENABLE_OFF);
        hstmtbusy[h] = FALSE;
        if (inserts)
                Get_Insert_Answer (h, rc);
        else
                Get_Delete_Answer (h, rc);

}



/* From print.c */
/*
 * NOTE: this routine does NOT use the BCD2_* routines. As a result,
 * it WILL fail if the keys being deleted exceed 32 bits. Since this
 * would require ~660 update iterations, this seems an acceptable
 * oversight
 */
int
gen_deletes (long min, long cnt, long num, int special)
{
        static long max_easy = LONG_MAX >> SPARSE_BITS;
```

```
        static int last_num = 0;
        static FILE *dfp = NULL;
        int child = -1;
        long new;
        struct timeb EndTime;
        struct timeb difTime;
        char tempStr[40];
        int rc;
        long i;
        long o;
        int h;

        BOOL anybusy;

        orderRowsDeleted = 0;
        orderRowsInError = 0;
        lineitemRowsDeleted = 0;
        orderRowsToBeDeleted = 0;

        if (special == 0)
                fprintf (fdetail, "\n Running update function UF2
(deletes)\n");
        else
                fprintf (fdetail, "\n Running special cleanup
deletes\n");
        if (flags & VERBOSE)
                {
                        if (special == 0)
                                fprintf (stdout, " Running update
function UF2 (deletes)\n");
                        else
                                fprintf (stdout, " Doing special cleanup
deletes\n");
                        //fprintf(stderr," Running update function UF2
(deletes)\n");
                }
        ftime (&StartTime);

        if (last_num != num)
                {
                        /* Do first time init here */
                        last_num = num;
                }
        gen_rng = 1;

        start = MK_SPARSE (min, special + num / (10000 / refresh));
        last = start - 1;
        h = 1;

        DR_Strt ();
        for (child = min; cnt > 0; child++, cnt--)
                {
                        if (child > max_easy)
                                {
                                        fprintf (stderr, " Deletes
failed:  Need to use mk_sparse\n");
                                        exit (1);
                                }
                                        new = MK_SPARSE (child, special + num / (10000 /
refresh));
                                        if (gen_rng == 1 && new - last == 1 && cnt > 1)
                                                {
                                                        last = new;
                                                        continue;
                                                }
                                        else
                                                {
                                                        if (cnt <= 1 && new - last == 1)
                                                                last = new;

                                                        SQLRequest[0] = '\0';
                                                        if (flags & EXPLAIN)
                                                                strcat (SQLRequest,
"EXPLAIN ");

                                                        //strcat(SQLRequest,"DELETE FROM
ORDERTBL WHERE O_ORDERKEY IN (");
                                                        OrdersInReq[h] = 0;
                                                        DR_Strt ();
                                                        strcat (SQLRequest, "USING (");
                                                        o = 0;
                                                        for (i = start; i <= last; i++)
                                                                {
                                                                        DR_Int (i);
                                                                        o++;
                                                                        strcat
(SQLRequest, "O");
                                                                        sprintf (tempStr,
"%ld", o);
                                                                        strcat
(SQLRequest, tempStr);
                                                                        strcat
(SQLRequest, " INT");
                                                                        if (i != last)
                                                                                strcat
(SQLRequest, ",");
                                                                }
                                                        if (cnt <= 1 && new > last)
                                                                {
                                                                        strcat
(SQLRequest, ",");
                                                                        DR_Int (new);
                                                                        o++;
                                                                        strcat
(SQLRequest, "O");
                                                                        sprintf (tempStr,
"%ld", o);
                                                                        strcat
(SQLRequest, tempStr);
                                                                        strcat
(SQLRequest, " INT");
                                                                }
                                                        strcat (SQLRequest, ") ");

                                                        o = 0;
                                                        for (i = start; i <= last; i++)
                                                                {
```

```
                                                o++;

        orderRowsToBeDeleted++;
                                                OrdersInReq[h]++;
                                                strcat
(SQLRequest, "DELETE FROM ORDERTBL WHERE O_ORDERKEY =:O");
                                                sprintf (tempStr,
"%ld", o);
                                                strcat
(SQLRequest, tempStr);
                                                //if (i!=last)
strcat(SQLRequest,",");
                                                strcat
(SQLRequest, ";");
                                        }
                                if (cnt <= 1 && new > last)
                                        {
                                                o++;

        orderRowsToBeDeleted++;
                                                OrdersInReq[h]++;
                                                strcat
(SQLRequest, "DELETE FROM ORDERTBL WHERE O_ORDERKEY =:O");
                                                sprintf (tempStr,
"%ld", o);
                                                strcat
(SQLRequest, tempStr);
                                                //if (i!=last)
strcat(SQLRequest,",");
                                                strcat
(SQLRequest, ";");
                                        }
                                //strcat(SQLRequest,");");

                                o = 0;
                                for (i = start; i <= last; i++)
                                        {
                                                o++;
                                                strcat
(SQLRequest, "DELETE FROM LINEITEM WHERE L_ORDERKEY =:O");
                                                sprintf (tempStr,
"%ld", o);
                                                strcat
(SQLRequest, tempStr);
                                                strcat
(SQLRequest, ";");
                                        }
                                if (cnt <= 1 && new > last)
                                        {
                                                o++;
                                                strcat
(SQLRequest, "DELETE FROM LINEITEM WHERE L_ORDERKEY =:O");
                                                sprintf (tempStr,
"%ld", o);
                                                strcat
(SQLRequest, tempStr);
                                                strcat
(SQLRequest, ";");
                                                }

                                        DR_End (LINE);

                                        Submit_Req (h, FALSE);
                                        Wait_For_Free_Hstmt (&h, FALSE);
                                        hstmt = hstmts[h];
                                        DR_Strt ();
                                        //fprintf(stderr," Next req goes
on hstmt %d\n",h);



                                        start = new;
                                        last = new;
                                }
                        }

                //fprintf(stderr," All submitted, waiting\n");
                anybusy = TRUE;
                while (anybusy)
                        {
                                anybusy = FALSE;
                                for (h = 1; h <= maxSess; h++)
                                        {
                                                if (hstmtbusy[h])
                                                        {
                                                                rc = SQLWaitForAny
(henv, &hstmt, (int *) &i);
                                                                if (rc ==
SQL_ERROR)
                                                                        {

                fprintf (stderr, "SQLWaitForAny returned SQL_ERROR\n");

                break;

                                                                        }
                                                                if (rc ==
SQL_NO_DATA_FOUND)
                                                                        break;

                                                                if (hstmts[i] !=
hstmt)
                                                                        fprintf
(stderr, "SQLWaitForAny returned garbage\n");


                //fprintf(stderr,"SQLWaitForAny returned %d\n",i);
                                                                Get_Delete_Answer
(i, (SQLRETURN) rc);
                                                        }
                                                if (hstmtbusy[h])
                                                        anybusy = TRUE;
                                        }
                        }
```

```
        hstmt = hstmts[1];
        ftime (&EndTime);
        difTime.time = EndTime.time - StartTime.time;
        if (EndTime.millitm >= StartTime.millitm)
                {
                        difTime.millitm = (unsigned short)
(EndTime.millitm - StartTime.millitm);
                }
        else
                {
                        difTime.millitm = (unsigned short) ((1000 +
EndTime.millitm) - StartTime.millitm);
                        difTime.time -= 1;
                };
        UF2time = difTime.time + difTime.millitm / 1000.0 + 0.005;

        StartTime.time = EndTime.time;
        StartTime.millitm = EndTime.millitm;

        if (orderRowsDeleted == 0 && special)
                UF2time = 0.0;
        else
                {
                        fprintf (fdetail, " UF2 Time was %.2f seconds.
Ended at %s\n\n", UF2time, ctime (&StartTime.time));


                        fprintf (fdetail, " Total %d deleted from Order
table\n", orderRowsDeleted);
                        if (orderRowsDeleted != orderRowsToBeDeleted)
                                fprintf (fdetail, " Should have been %d
deleted from Order table!!\n", orderRowsToBeDeleted);
                        fprintf (fdetail, " Total %d deleted from
LineItem table\n", lineitemRowsDeleted);
                        if (orderRowsInError > 0)
                                fprintf (fdetail, " Total %d deletes
failed due to database errors\n", orderRowsInError);

                }



        return (0);
}


int
gen_deletes_via_fastload (long min, long cnt, long num, int special)
{
        static long max_easy = LONG_MAX >> SPARSE_BITS;
        static int last_num = 0;
        static FILE *dfp = NULL;
        int child = -1;

        char SqlState[6];
        char ErrMsg[256];
        short ErrMsgLen;
        long ErrCode;
        BOOL retry;

        long new;
        struct timeb EndTime;
        struct timeb difTime;


        int h;



        orderRowsDeleted = 0;
        lineitemRowsDeleted = 0;
        orderRowsToBeDeleted = 0;

        prep_direct((db_name)?db_name:DBNAME);


        if (special == 0)
                fprintf (fdetail, "\n Running update function UF2
(deletes)\n");
        else
                fprintf (fdetail, "\n Running special cleanup
deletes\n");
        if (flags & VERBOSE)
                {
                        if (special == 0)
                                fprintf (stdout, " Running update
function UF2 (deletes)\n");
                        else
                                fprintf (stdout, " Doing special cleanup
deletes\n");
                        //fprintf(stderr," Running update function UF2
(deletes)\n");
                }
        ftime (&StartTime);

        /* In this version of UF2, we first use the Teradata "fastload"
utility
           to send the keys to be deleted to a temporary table on the
Teradata system.
           After all the rows have been sent, we use two DELETE
statements
           to delete the rows from the existing tables */

        if (last_num != num)
                {
                        /* Do first time init here */
                        last_num = num;
                }
        gen_rng = 1;

        start = MK_SPARSE (min, special + num / (10000 / refresh));
        last = start - 1;
        h = 1;

        DR_Strt ();
        for (child = min; cnt > 0; child++, cnt--)
```

```
		{
			if (child > max_easy)
				{
					fprintf (stderr, " Deletes
failed:  Need to use mk_sparse\n");
					exit (1);
				}

			new = MK_SPARSE (child, special + num / (10000 /
refresh));
			ld_deletes (new, upd_num);
			orderRowsToBeDeleted++;

		}

	close_direct();



	hstmt = hstmts[1];


	/* By doing both DELETE statements in a single request to the
Teradata system,
	   Teradata guarantees these are done as an atomic transaction
*/

	retry = TRUE;
	while (retry)
	{
		retry = FALSE;
		if (SQLExecDirect (hstmt, (unsigned char *)
	"DELETE FROM LINEITEM WHERE L_ORDERKEY =
KEYS_TO_DELETE.ORDERKEY;DELETE FROM ORDERTBL WHERE O_ORDERKEY =
KEYS_TO_DELETE.ORDERKEY",
			SQL_NTS) == SQL_ERROR)
			{
				fprintf (fdetail, " UF2 delete
failed\n");
				SQLError ((SQLHENV) 0, (SQLHDBC) 0,
hstmt, (SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256,
&ErrMsgLen);
				fprintf (fdetail, " %d: %s\n", ErrCode,
ErrMsg);
				fprintf (stdout, " UF2 delete failed\n");
				fprintf (stdout, " %d: %s\n", ErrCode,
ErrMsg);
				SQLFreeStmt (hstmt, SQL_CLOSE);
				if (canRetry(ErrCode)) /* Transaction
ABORTed due to DeadLock or crash */
					{
						retry = TRUE;
						fprintf (fdetail, "
Attempting to resubmit UF2 delete\n");
						fprintf (stdout, "
Attempting to resubmit UF2 delete\n");
					}
			}
		else
			{
				if (SQLRowCount (hstmt,
&lineitemRowsDeleted) != SQL_SUCCESS)
					fprintf (stderr, " SQLRowCount
problem\n");

				if (SQLMoreResults (hstmt) !=
SQL_SUCCESS)
					fprintf (stderr, " SQLMoreResults
problem\n");

				if (SQLRowCount (hstmt,
&orderRowsDeleted) != SQL_SUCCESS)
					fprintf (stderr, " SQLRowCount
problem\n");
			}
	}


	ftime (&EndTime);
	difTime.time = EndTime.time - StartTime.time;
	if (EndTime.millitm >= StartTime.millitm)
		{
			difTime.millitm = (unsigned short)
(EndTime.millitm - StartTime.millitm);
		}
	else
		{
			difTime.millitm = (unsigned short) ((1000 +
EndTime.millitm) - StartTime.millitm);
			difTime.time -= 1;
		};
	UF2time = difTime.time + difTime.millitm / 1000.0 + 0.005;

	StartTime.time = EndTime.time;
	StartTime.millitm = EndTime.millitm;

	if (orderRowsDeleted == 0 && special)
		UF2time = 0.0;
	else
		{
			fprintf (fdetail, " UF2 Time was %.2f seconds.
Ended at %s\n\n", UF2time, ctime (&StartTime.time));


			fprintf (fdetail, " Total %d deleted from Order
table\n", orderRowsDeleted);
			if (orderRowsDeleted != orderRowsToBeDeleted)
				fprintf (fdetail, " Should have been %d
deleted from Order table!!\n", orderRowsToBeDeleted);
			fprintf (fdetail, " Total %d deleted from
LineItem table\n", lineitemRowsDeleted);

		}
```

```
        return (0);
}



int
Build_Multistatement_Insert (int h)
{
        char tstr2[6] = "?";
        char tstr1[3] = "?";
        char tstr3[4] = "?";
        int j;
        int k;

        /*if ((flags & VERBOSE)&&(specialtest))
           fprintf(fdetail," Building multi insert, orders %d, lines[0]
=%d\n",OrdersInReq,LinesPerOrder[0]);
         */
        SQLRequest[0] = '\0';
        if (flags & EXPLAIN)
                strcpy (SQLRequest, " EXPLAIN ");
        sprintf (tstr2, "%d", LinesPerOrder[h][1] + 1);
        strcat (SQLRequest, "/*");
        strcat (SQLRequest, tstr2);
        strcat (SQLRequest, "*/ ");
        strcat (SQLRequest, "USING ");

        if (OrdersInReq[h] > 10 || OrdersInReq[h] > setsPerReq ||
OrdersInReq[h] <= 0)
                {
                        fprintf (stderr, "OrdersInReq[h] = %d\n",
OrdersInReq[h]);
                        fclose (fdetail);
                        fclose (fsummary);
                        exit (1);
                }

        // First, build the using info for all the ORDER inserts
        for (k = 0; k < OrdersInReq[h]; k++)
        {
                        tstr1[0] = (char) ('0' + k);
                        tstr1[1] = '\0';
                        if (k > 0)
                                strcat (SQLRequest, ",");
                        strcat (SQLRequest, "OOK");
                        strcat (SQLRequest, tstr1);
                        if (scale > 300)
                                strcat (SQLRequest, "(DEC(15,0))");
                        else
                                strcat (SQLRequest, "(INT)");
                        strcat (SQLRequest, ",OCK");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, "(INT)");
                        strcat (SQLRequest, ",OOS");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, "(CHAR(1))");
                        strcat (SQLRequest, ",OTP");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, "(DEC(15,2))");
                        strcat (SQLRequest, ",OOD");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, "(DATE)");
                        strcat (SQLRequest, ",OOP");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, "(CHAR(15))");
                        strcat (SQLRequest, ",OC");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, "(CHAR(15))");
                        strcat (SQLRequest, ",OSP");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, "(INT)");
                        strcat (SQLRequest, ",OCM");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, "(VARCHAR(79)) ");
                        if (LinesPerOrder[h][k] > 20)
                                {
                                        fprintf (stderr, "
LinesPerOrder[h][k] > 20, %d\n", LinesPerOrder[h][k]);
                                        fclose (fdetail);
                                        fclose (fsummary);
                                        exit (1);
                                }

        }

        // Now build all the using data for the lineitems.
        for (j = 0; j < 20; j++)
                for (k = 0; k < OrdersInReq[h]; k++)
                {
                        if (j < LinesPerOrder[h][k])
                                {
                                        tstr1[0] = (char) ('0' + k);
                                        tstr1[1] = '\0';
                                        tstr3[0] = (char) ('0' + j / 10);
                                        tstr3[1] = (char) ('0' + j % 10);
                                        tstr3[2] = '\0';
                                        strcpy (tstr2, tstr1);
                                        strcat (tstr2, tstr3);

                                        strcat (SQLRequest, ",LOK");
                                        strcat (SQLRequest, tstr2);
                                        if (scale > 300)

                                                strcat (SQLRequest,
"(DEC(15,0))");
                                        else
                                                strcat (SQLRequest,
"(INT)");

                                        strcat (SQLRequest, ",LPK");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, "(INT)");

                                        strcat (SQLRequest, ",LSK");
                                        strcat (SQLRequest, tstr2);
```

```
                                        strcat (SQLRequest, "(INT)");
                                        strcat (SQLRequest, ",LLN");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, "(INT)");
                                        strcat (SQLRequest, ",LQ");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, "(INT)");
                                        strcat (SQLRequest, ",LEP");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest,
"(DEC(15,2))");
                                        strcat (SQLRequest, ",LDSC");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest,
"(DEC(15,2))");
                                        strcat (SQLRequest, ",LTAX");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest,
"(DEC(15,2))");
                                        strcat (SQLRequest, ",LRTNF");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, "(CHAR(1))");
                                        strcat (SQLRequest, ",LLS");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, "(CHAR(1))");
                                        strcat (SQLRequest, ",LSHD");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, "(DATE)");
                                        strcat (SQLRequest, ",LCMTD");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, "(DATE)");
                                        strcat (SQLRequest, ",LRCD");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, "(DATE)");
                                        strcat (SQLRequest, ",LSHI");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest,
"(CHAR(25))");
                                        strcat (SQLRequest, ",LSHM");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest,
"(CHAR(10))");
                                        strcat (SQLRequest, ",LCM");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest,
"(VARCHAR(44)) ");

                                }
                        }

        // Next, build the inserts for all the ORDER inserts.
        for (k = 0; k < OrdersInReq[h]; k++)
                {
                        tstr1[0] = (char) ('0' + k);
                        tstr1[1] = '\0';
                        strcat (SQLRequest, " INSERT INTO ORDERTBL
VALUES(");
                        strcat (SQLRequest, ":OOK");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, ",:OCK");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, ",:OOS");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, ",:OTP");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, ",:OOD");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, ",:OOP");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, ",:OC");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, ",:OSP");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, ",:OCM");
                        strcat (SQLRequest, tstr1);
                        strcat (SQLRequest, ");");
                }
        for (j = 0; j < 20; j++)
                for (k = 0; k < OrdersInReq[h]; k++)
                {
                        if (j < LinesPerOrder[h][k])
                                {
                                        tstr1[0] = (char) ('0' + k);
                                        tstr1[1] = '\0';
                                        tstr3[0] = (char) ('0' + j / 10);
                                        tstr3[1] = (char) ('0' + j % 10);
                                        tstr3[2] = '\0';
                                        strcpy (tstr2, tstr1);
                                        strcat (tstr2, tstr3);

                                        strcat (SQLRequest, " INSERT INTO
LINEITEM VALUES(");
                                        strcat (SQLRequest, ":LOK");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ",:LPK");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ",:LSK");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ",:LLN");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ",:LQ");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ",:LEP");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ",:LDSC");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ",:LTAX");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ",:LRTNF");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ",:LLS");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ",:LSHD");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ",:LCMTD");
                                        strcat (SQLRequest, tstr2);
```

```
                                        strcat (SQLRequest, ",:LRCD");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ",:LSHI");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ",:LSHM");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ",:LCM");
                                        strcat (SQLRequest, tstr2);
                                        strcat (SQLRequest, ");");

                                }
                }
        return (0);
}



/*
 * generate a particular table
 */
void
gen_updates (long start, long count, long upd_num)
{
        long i;

        int h;

        long sk_low, sk_high;
        static long max_easy = LONG_MAX >> SPARSE_BITS;
        static int completed = 0;

        struct timeb EndTime;
        struct timeb difTime;
        int rc;
        long maxReqLen = 0;
        long maxUsingLen = 0;

        BOOL anybusy;

        orderRowsAdded = 0;
        orderRowsInError = 0;
        lineitemRowsAdded = 0;
        orderRowsToBeAdded = 0;
        lineitemRowsToBeAdded = 0;
        duplicates = 0;

        fprintf (fdetail, "\n Running update function UF1
(inserts)\n");
        if (flags & VERBOSE)
                {
                        fprintf (stdout, " Running update function UF1
(inserts)\n");
                        //fprintf(stderr," Running update function UF1
(inserts)\n");
                }
        ftime (&StartTime);


        for (i = 1; i <= maxSess; i++)
                {
                        hstmtbusy[i] = FALSE;
                        OrdersInReq[i] = 0;
                        LinesPerOrder[i][0] = 0;
                        LinesPerOrder[i][1] = 0;
                }

        h = 1;
        hstmt = hstmts[1];

        DR_Strt ();
        for (i = start; count; count--, i++)
                {
                        LIFENOISE (1000);

                        if (i > max_easy)
                                mk_sparse (&sk_low, &sk_high, i,
                                                1 + upd_num / (10000 /
refresh));
                        else
                                {
                                        sk_low = MK_SPARSE (i,

        1 + upd_num / (10000 / refresh));
                                        sk_high = 0;
                                }


                        mk_order (sk_low, sk_high,
&o[h][OrdersInReq[h]]);

                        LinesPerOrder[h][OrdersInReq[h]] =
o[h][OrdersInReq[h]].lines;
                        orderRowsToBeAdded++;
                        lineitemRowsToBeAdded +=
o[h][OrdersInReq[h]].lines;

                        OrdersInReq[h]++;
                        LinesPerOrder[h][OrdersInReq[h]] = 0;


                        if ((OrdersInReq[h] >= setsPerReq) || (count <=
1))
                                {
                                        int k;
                                        int j;
                                        Build_Multistatement_Insert (h);
                                        for (k = 0; k < OrdersInReq[h];
k++)
                                                xld_order (&o[h][k],
upd_num);

                                        for (j = 0; j < 20; j++)
                                                for (k = 0; k <
OrdersInReq[h]; k++)
```

```
                                                if (j <
LinesPerOrder[h][k])

        xld_one_line (&o[h][k], j, upd_num);

                                        if (strlen (SQLRequest) >
(size_t) maxReqLen)
                                                maxReqLen = strlen
(SQLRequest);
                                        if (((int) (p1 - UsingBuffer)) >
maxUsingLen)
                                                maxUsingLen = (int) (p1 -
UsingBuffer);



                                        Submit_Req (h, TRUE);
                                        Wait_For_Free_Hstmt (&h, TRUE);
                                        hstmt = hstmts[h];
                                        DR_Strt ();
                                        OrdersInReq[h] = 0;
                                        //fprintf(stderr," Next req goes
on hstmt %d\n",h);
                                }
                }

        //fprintf(stderr," All submitted, waiting\n");
        anybusy = TRUE;
        while (anybusy)
                {
                        anybusy = FALSE;
                        for (h = 1; h <= maxSess; h++)
                                {
                                        if (hstmtbusy[h])
                                                {
                                                        rc = SQLWaitForAny
(henv, &hstmt, (int *) &i);
                                                        if (rc ==
SQL_ERROR)
                                                                {

        fprintf (stderr, "SQLWaitForAny returned SQL_ERROR\n");

        break;
                                                                }
                                                        if (rc ==
SQL_NO_DATA_FOUND)
                                                                break;

                                                        if (hstmts[i] !=
hstmt)
                                                                fprintf
(stderr, "SQLWaitForAny returned garbage\n");


        //fprintf(stderr,"SQLWaitForAny returned %d\n",i);
                                                        Get_Insert_Answer
(i, (SQLRETURN) rc);
                                                }
                                        if (hstmtbusy[h])
                                                anybusy = TRUE;
                                }
                        }

        hstmt = hstmts[1];

        ftime (&EndTime);
        difTime.time = EndTime.time - StartTime.time;
        if (EndTime.millitm >= StartTime.millitm)
                {
                        difTime.millitm = (unsigned short)
(EndTime.millitm - StartTime.millitm);
                }
        else
                {
                        difTime.millitm = (unsigned short) ((1000 +
EndTime.millitm) - StartTime.millitm);
                        difTime.time -= 1;
                };
        UF1time = difTime.time + difTime.millitm / 1000.0 + 0.005;

        StartTime.time = EndTime.time;
        StartTime.millitm = EndTime.millitm;

        fprintf (fdetail, " UF1 Insert Time was %.2f seconds.  Ended at
%s\n\n", UF1time, ctime (&StartTime.time));

#ifdef _DEBUG
        if (flags & VERBOSE)
                fprintf (stdout, " Insert stmt max len %d, using len
%d\n", maxReqLen, maxUsingLen);
#endif

        fprintf (fdetail, " Total %d inserted into Order table\n",
orderRowsAdded);
        if (orderRowsAdded != orderRowsToBeAdded)
                fprintf (fdetail, " Should have been %d inserted into
Order table!!\n", orderRowsToBeAdded);
        fprintf (fdetail, " Total %d inserted into LineItem table\n",
lineitemRowsAdded);
        if (lineitemRowsAdded != lineitemRowsToBeAdded)
                fprintf (fdetail, " Should have been %d inserted into
LineItem table\n", lineitemRowsToBeAdded);

        if ((flags & VERBOSE) || (duplicates > 0))
                fprintf (fdetail, " Total %d inserts ignored due to
duplicate unique primary index\n", duplicates);

        if (orderRowsInError > 0)
                fprintf (fdetail, " Total %d inserts failed due to
database errors\n", orderRowsInError);
}

/*
 * generate a particular table
 */
```

```
void
gen_updates_via_fastload (long start, long count, long upd_num)
{
        long i;

        long sk_low, sk_high;
        static long max_easy = LONG_MAX >> SPARSE_BITS;
        static int completed = 0;
        order_t o;

        char SqlState[6];
        char ErrMsg[256];
        short ErrMsgLen;
        long ErrCode;
        BOOL retry;

        struct timeb EndTime;
        struct timeb difTime;


        orderRowsAdded = 0;
        lineitemRowsAdded = 0;
        orderRowsToBeAdded = 0;
        lineitemRowsToBeAdded = 0;


        prep_direct((db_name)?db_name:DBNAME);


        fprintf (fdetail, "\n Running update function UF1
(inserts)\n");
        if (flags & VERBOSE)
                {
                        fprintf (stdout, " Running update function UF1
(inserts)\n");
                        //fprintf(stderr," Running update function UF1
(inserts)\n");
                }
        ftime (&StartTime);

        /* In this version of UF1, we first use the Teradata "fastload"
utility
           to send the rows to be inserted to temporary tables on the
Teradata system.
           After all the rows have been sent, we use INSERT/SELECT
statements
           to merge the new rows with the existing tables */

        for (i = start; count; count--, i++)
                {
                        LIFENOISE (1000);

                        if (i > max_easy)
                                mk_sparse (&sk_low, &sk_high, i,
                                                1 + upd_num / (10000 /
refresh));
                        else
                                {
                                        sk_low = MK_SPARSE (i,
        1 + upd_num / (10000 / refresh));
                                        sk_high = 0;
                                }


                        mk_order (sk_low, sk_high, &o);

                        orderRowsToBeAdded++;
                        lineitemRowsToBeAdded += o.lines;


                        ld_order_line (&o, upd_num);



                }


    close_direct();

        hstmt = hstmts[1];

        /* By doing both INSERT/SELECT statements in a single request
to the Teradata system,
           Teradata guarantees these are done as an atomic transaction
*/

        retry = TRUE;
        while (retry)
        {
                retry = FALSE;
                if (SQLExecDirect (hstmt, (unsigned char *)
        "INSERT INTO ORDERTBL SELECT * FROM ORDERTBL_UPDATES;INSERT
INTO LINEITEM SELECT * FROM LINEITEM_UPDATES",
                        SQL_NTS) == SQL_ERROR)
                        {
                                fprintf (fdetail, " UF1 Insert/select
failed\n");
                                SQLError ((SQLHENV) 0, (SQLHDBC) 0,
hstmt, (SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256,
&ErrMsgLen);
                                fprintf (fdetail, " %d: %s\n", ErrCode,
ErrMsg);
                                fprintf (stdout, " UF1 Insert/select
failed\n");
                                fprintf (stdout, " %d: %s\n", ErrCode,
ErrMsg);
                                SQLFreeStmt (hstmt, SQL_CLOSE);
                                if (canRetry(ErrCode)) /* Transaction
ABORTed due to DeadLock or crash */
                                        {
                                                retry = TRUE;
                                                fprintf (fdetail, "
Attempting to resubmit UF1 Insert/select\n");
                                                fprintf (stdout, "
Attempting to resubmit UF1 Insert/select\n");
```

```
                                        }
                        }
                else
                        {
                                if (SQLRowCount (hstmt,
&lineitemRowsAdded) != SQL_SUCCESS)
                                        fprintf (stderr, " SQLRowCount
problem\n");

                                if (SQLMoreResults (hstmt) !=
SQL_SUCCESS)
                                        fprintf (stderr, " SQLMoreResults
problem\n");

                                if (SQLRowCount (hstmt, &orderRowsAdded)
!= SQL_SUCCESS)
                                        fprintf (stderr, " SQLRowCount
problem\n");
                        }
        }


        ftime (&EndTime);
        difTime.time = EndTime.time - StartTime.time;
        if (EndTime.millitm >= StartTime.millitm)
                {
                        difTime.millitm = (unsigned short)
(EndTime.millitm - StartTime.millitm);
                }
        else
                {
                        difTime.millitm = (unsigned short) ((1000 +
EndTime.millitm) - StartTime.millitm);
                        difTime.time -= 1;
                };
        UF1time = difTime.time + difTime.millitm / 1000.0 + 0.005;

        StartTime.time = EndTime.time;
        StartTime.millitm = EndTime.millitm;

        fprintf (fdetail, " UF1 Insert Time was %.2f seconds.  Ended at
%s\n\n", UF1time, ctime (&StartTime.time));

        fprintf (fdetail, " Total %d inserted into Order table\n",
orderRowsAdded);
        if (orderRowsAdded != orderRowsToBeAdded)
                fprintf (fdetail, " Should have been %d inserted into
Order table!!\n", orderRowsToBeAdded);
        fprintf (fdetail, " Total %d inserted into LineItem table\n",
lineitemRowsAdded);
        if (lineitemRowsAdded != lineitemRowsToBeAdded)
                fprintf (fdetail, " Should have been %d inserted into
LineItem table\n", lineitemRowsToBeAdded);


}
```

```
void
stop_proc (int signum)
{
        fflush (fdetail);
        fflush (fsummary);
        fflush (stdout);
        fprintf (stderr, " Program terminating abnormally!
signal=%d\n", signum);
        fflush (stderr);
        exit (1);
}


int
Connect_Sessions ()
{
        char SqlState[6];
        char ErrMsg[256];
        short ErrMsgLen;
        long ErrCode;

        int i;
        char szLogon[240];
        SQLCHAR szDSN[32];
        SQLCHAR szUID[32];
        SQLCHAR szPWD[32];
        char tempstr[100];

        char *place;
        char *place2;

        szDSN[0] = '\0';
        szUID[0] = '\0';
        szPWD[0] = '\0';
        szLogon[0] = '\0';

        strcpy (szLogon, db_name);


        if (strstr (szLogon, "/") != NULL)
                {

                        place = strstr (szLogon, "/");
                        place[0] = '\0';
                        strcpy ((char *) szDSN, szLogon);
                        place++;
                        place2 = strstr (place, ",");
                        place2[0] = '\0';
                        strcpy ((char *) szUID, place);
                        place2++;
                        strcpy ((char *) szPWD, place2);

                }

        if (snum == 0)
                {
                        sprintf (tempstr, "rsh %scop1
/tpasw/bin/fsuflusher", szDSN);
```

```
                        if (system (tempstr) != 0)
                                {
                                        fprintf (stderr, " Tried to '%s'
but it didn't work\n", tempstr);
                                        sleep (5);
                                }
                        else
                                {
                                        fprintf (fdetail, " Called
fsuflusher to flush memory\n");
                                        fprintf (stdout, " Called
fsuflusher to flush memory\n");
                                        sleep (15);
                                }
                }


        if (SQLAllocEnv (&henv) != SQL_SUCCESS)
                {
                        fprintf (stderr, "Could not allocate ENV\n");
                        exit (1);
                }

        for (i = 1; i <= maxSess; i++)
                {
                        if (SQLAllocConnect (henv, &hdbc) !=
SQL_SUCCESS)
                                {
                                        fprintf (stderr, "Could not
allocate DBC\n");
                                        exit (1);
                                }
                        if (SQLConnect (hdbc, szDSN, SQL_NTS, szUID,
SQL_NTS, szPWD, SQL_NTS) != SQL_SUCCESS)
                                {
                                        if (i > 1)
                                                fprintf (stderr, " When
trying to log on the %dth session, \n", i);
                                        fprintf (stderr, "Could not log
on using '%s/%s,%s'\n", szDSN, szUID, szPWD);
                                        SQLError ((SQLHENV) 0, hdbc,
(SQLHSTMT) 0, (SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256,
&ErrMsgLen);
                                        fprintf (stderr, " %s (%ld)
%s\n", SqlState, ErrCode, ErrMsg);
                                        if (i < 10)
                                                exit (1);
                                        fprintf (stderr, " Continuing
with %d sessions only\n", i - 1);
                                        fflush (stderr);
                                        maxSess = i - 1;
                                        hdbc = hdbcs[1];
                                        hstmt = hstmts[1];

                                        return (0);

                                }
                        if (SQLAllocStmt (hdbc, &hstmt) != SQL_SUCCESS)
                                {
                                        fprintf (stderr, "Could not
allocate STMT\n");
                                        exit (1);
                                }
                        hdbcs[i] = hdbc;
                        hstmts[i] = hstmt;
                        if (updates == 0)
                                break;

                }
        hdbc = hdbcs[1];
        hstmt = hstmts[1];

        return (0);
}
int
Disconnect_Sessions ()
{
        int i;
        for (i = 1; i <= maxSess; i++)
                {
                        SQLFreeStmt (hstmts[i], SQL_DROP);
                        hstmt = NULL;
                        if (SQLDisconnect (hdbcs[i]) == SQL_ERROR)
                                fprintf (stderr, "Error when logging off
the sessions\n");
                        if (updates == 0)
                                break;
                }

        if (flags & VERBOSE)
                fprintf (stdout, "Sessions are now logged off from
Teradata\n");
        return (0);
}

int
Run_Query (const char *SQLReq, int qnum)
{
        static char SQLR[2048];
        char SqlState[6];
        char ErrMsg[256];
        short ErrMsgLen;
        long ErrCode;
        BOOL retry;

        struct timeb EndTime;
        struct timeb difTime;
        int i;
        char *SQ1;
        char *SQ2;
        const char *SQ;
        strcpy (SQLR, SQLReq);

        fprintf (fdetail, "\n Submitting SQL Request #%d:\n\n", qnum);
        fprintf (stdout, " Submitting SQL Request #%d:\n", qnum);
```

```
        SQ2 = SQLR;

        while (SQ2 != NULL)
                {
                        SQ1 = SQ2;
                        for (i = strlen (SQ1) - 1; i > 0; i--)       /*
Trim trailing spaces */
                                if (SQ1[i] == ' ' ||
                                        SQ1[i] == '\r' ||
                                        SQ1[i] == '\t')
                                        SQ1[i] = '\0';
                                else
                                        break;
                        if (SQ1[strlen (SQ1) - 1] == ';')     /* Trim
trailing semicolon and spaces */
                                {
                                        SQ1[strlen (SQ1) - 1] = '\0';
                                        for (i = strlen (SQ1) - 1; i > 0;
i--)
                                                if (SQ1[i] == ' ' ||
                                                        SQ1[i] == '\r' ||
                                                        SQ1[i] == '\t')
                                                        SQ1[i] = '\0';
                                                else
                                                        break;
                                }

                        if (strstr (SQ1, "/*") != NULL)
                                SQ2 = strstr (strstr (SQ1, "/*"),
"*/"), ";");   /* look past the comment strings! */
                        else
                                SQ2 = strstr (SQ1, ";");      /* If
semicolon, this is a multi-statement request */
                        if (SQ2 != NULL)
                                {
                                        *SQ2 = '\0';   /* Break out the
first SQL statement in SQ1 */
                                        SQ2++;         /* SQ2 points to
the next statement */
                                }

                        SQ = SQ1;
                        while (SQ[0] != 0)
                                {

                                        if (SQ[0] == '\r')
                                                fputc ('\n', fdetail);
                                        else
                                                fputc (SQ[0], fdetail);
                                        SQ++;
                                }
                        fprintf (fdetail, ";\n\n");

                        retry = TRUE;
                        while (retry)
                        {
                                retry = FALSE;
                                if (SQLExecDirect (hstmt, (unsigned char
*) SQ1, SQL_NTS) == SQL_ERROR)
                                        {
                                                fprintf (fdetail, " Query
#%d failed\n", qnum);
                                                SQLError ((SQLHENV) 0,
(SQLHDBC) 0, hstmt, (SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg,
256, &ErrMsgLen);
                                                fprintf (fdetail, " %d:
%s\n", ErrCode, ErrMsg);
                                                fprintf (stdout, " Query
#%d failed\n", qnum);
                                                fprintf (stdout, " %d:
%s\n", ErrCode, ErrMsg);
                                                SQLFreeStmt (hstmt,
SQL_CLOSE);
                                                if (canRetry(ErrCode)) /*
Transaction ABORTed due to DeadLock or crash */
                                                        {
                                                                retry =
TRUE;
                                                                fprintf
(fdetail, " Attempting to resubmit Query #%d\n", qnum);
                                                                fprintf
(stdout, " Attempting to resubmit Query #%d\n", qnum);
                                                        }
                                                else
                                                if (ErrCode != 3803)   /*
table already exists */
                                                        {

        QueryTime[qnum] = 0.0;
                                                                return (-
1);
                                                        }
                                        }
                                else
                                        {
                                                Fetch_and_Display_All_Rows
(hstmt, qnum);
                                        }
                        }
                }

        ftime (&EndTime);
        difTime.time = EndTime.time - StartTime.time;
        if (EndTime.millitm >= StartTime.millitm)
                {
                        difTime.millitm = (unsigned short)
(EndTime.millitm - StartTime.millitm);
                }
        else
                {
                        difTime.millitm = (unsigned short) ((1000 +
EndTime.millitm) - StartTime.millitm);
                        difTime.time -= 1;
                };
```

```
        QueryTime[qnum] = difTime.time + difTime.millitm / 1000.0 +
0.005;

        StartTime.time = EndTime.time;
        StartTime.millitm = EndTime.millitm;

        fprintf (fdetail, "Time was %.2f seconds.  Query ended at
%s\n\n", QueryTime[qnum], ctime (&StartTime.time));



        return (0);

}


int
Verify_Database ()
{
        long actualLen;
        long crowcount;
        double newSF;
        char SqlState[6];
        char ErrMsg[256];
        char InfoData[256];
        short ErrMsgLen;
        long ErrCode;
        long vAMPs;
        int isV2;

        isV2 = 0;
        if (SQLExecDirect (hstmt, (SQLCHAR *) "SELECT INFODATA FROM
DBC.DBCINFO WHERE INFOKEY='RELEASE'", SQL_NTS) != SQL_ERROR)
                {
                        if (SQLFetch (hstmt) == SQL_SUCCESS)
                                if (SQLGetData (hstmt, 1, SQL_C_CHAR,
InfoData, 256, &actualLen) == SQL_SUCCESS)
                                        {
                                                fprintf (fdetail, " The
system under test is running Teradata release %s\n", InfoData);
                                                fprintf (stdout, " The
system under test is running Teradata release %s\n", InfoData);
                                                if (InfoData[0] == 'V' &&
InfoData[1] == '2')
                                                        isV2 = 1;
                                        }
                        SQLFreeStmt (hstmt, SQL_CLOSE);
                }

        vAMPs = 0;
        if (isV2)
                if (SQLExecDirect (hstmt, (SQLCHAR *) "SELECT
HASHAMP(HASHBUCKET(HASHROW(S_SUPPKEY))) FROM SUPPLIER GROUP BY 1 ORDER
BY 1", SQL_NTS) != SQL_ERROR)
                        {
                                if (SQLRowCount (hstmt, &vAMPs) !=
SQL_SUCCESS)
                                        fprintf (stderr, " Problem in
SQLRowCount!\n");
                                if (vAMPs > 0)
                                        {
                                                fprintf (fdetail, " The
system under test has %d virtual AMPs\n", vAMPs);
                                                fprintf (stdout, " The
system under test has %d virtual AMPs\n", vAMPs);
                                        }

                                SQLFreeStmt (hstmt, SQL_CLOSE);
                        }

        if (SQLExecDirect (hstmt, (SQLCHAR *) "SELECT COUNT(*) FROM
DBC.SESSIONINFO", SQL_NTS) != SQL_ERROR)
                {
                        if (SQLFetch (hstmt) == SQL_SUCCESS)
                                if (SQLGetData (hstmt, 1, SQL_C_SLONG,
&crowcount, 4, &actualLen) == SQL_SUCCESS)
                                        {
                                                if (crowcount > maxSess)
                                                        {
                                                                fprintf
(fdetail, " There are %d other sessions logged on to the system under
test\n", crowcount - maxSess);
                                                                fprintf
(stdout, " There are %d other sessions logged on to the system under
test\n", crowcount - maxSess);
                                                        }
                                        }
                        SQLFreeStmt (hstmt, SQL_CLOSE);
                }


        if (flags & VERBOSE)
                fprintf (stdout, "Submitting SQL Requests to verify
scale factor\n");
        if (SQLExecDirect (hstmt, (SQLCHAR *) "SELECT COUNT(*) FROM
CUSTOMER", SQL_NTS) != SQL_ERROR)
                {
                        if (SQLFetch (hstmt) == SQL_SUCCESS)
                                if (SQLGetData (hstmt, 1, SQL_C_SLONG,
&crowcount, 4, &actualLen) == SQL_SUCCESS)
                                        {
                                                SQLFreeStmt (hstmt,
SQL_CLOSE);
                                                newSF = ((double)
crowcount) / 150000.0;
                                                fprintf (stdout, "
Customer table exists with scale factor %g\n", newSF);
                                                if ((long) (newSF * 1000 +
0.5) != (long) (flt_scale * 1000))
                                                        fprintf (stdout, "
Scale factor does not match!\n");
                                        }
                }
        else
                {
```

```
                        fprintf (stderr, " Customer table does not exist
or cant be accessed\n");
                        SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmt,
(SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256, &ErrMsgLen);
                        fprintf (stderr, " %d: %s\n", ErrCode, ErrMsg);
                        SQLFreeStmt (hstmt, SQL_CLOSE);
                }

        if (SQLExecDirect (hstmt, (SQLCHAR *) "SELECT COUNT(*) FROM
PARTTBL", SQL_NTS) != SQL_ERROR)
                {
                        if (SQLFetch (hstmt) == SQL_SUCCESS)
                                if (SQLGetData (hstmt, 1, SQL_C_SLONG,
&crowcount, 4, &actualLen) == SQL_SUCCESS)
                                        {
                                                SQLFreeStmt (hstmt,
SQL_CLOSE);
                                                newSF = ((double)
crowcount) / 200000.0;
                                                fprintf (stdout, " Part
table exists with scale factor %g\n", newSF);
                                                if ((long) (newSF * 1000 +
0.5) != (long) (flt_scale * 1000))
                                                        fprintf (stdout, "
Scale factor does not match!\n");
                                        }
                }
        else
                {
                        fprintf (stderr, " Part table does not
exist\n");
                        SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmt,
(SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256, &ErrMsgLen);
                        fprintf (stderr, " %d: %s\n", ErrCode, ErrMsg);
                        SQLFreeStmt (hstmt, SQL_CLOSE);
                }


        if (SQLExecDirect (hstmt, (SQLCHAR *) "SELECT COUNT(*) FROM
ORDERTBL", SQL_NTS) != SQL_ERROR)
                {
                        if (SQLFetch (hstmt) == SQL_SUCCESS)
                                if (SQLGetData (hstmt, 1, SQL_C_SLONG,
&crowcount, 4, &actualLen) == SQL_SUCCESS)
                                        {
                                                SQLFreeStmt (hstmt,
SQL_CLOSE);
                                                newSF = ((double)
crowcount) / 1500000.0;
                                                fprintf (stdout, " Order
table exists with scale factor %g\n", newSF);
                                                if ((long) (newSF * 1000 +
0.5) != (long) (flt_scale * 1000))
                                                        fprintf (stdout, "
Scale factor does not match!\n");
                                        }
                }
        else
                {
                        fprintf (stderr, " Order table does not
exist\n");
                        SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmt,
(SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256, &ErrMsgLen);
                        fprintf (stderr, " %d: %s\n", ErrCode, ErrMsg);
                        SQLFreeStmt (hstmt, SQL_CLOSE);
                }


        return (0);

}

int
Run_Query_Stream ()
{
        int i;
        ftime (&StartTime);
        fprintf (stdout, " Starting query run for stream %d on %s\n",
snum, ctime (&StartTime.time));
        fprintf (fdetail, "\n Starting query run for stream %d on
%s\n", snum, ctime (&StartTime.time));

        ftime (&StartTime);



        if (snum >= 0)
                if (optind < ac1)
                        for (i = optind; i < ac1; i++)
                                {
                                        char qname[10];
                                        sprintf (qname, "%d", SEQUENCE
(snum, atoi (av1[i])));
                                        qsub (qname, flags);
                                        Run_Query (SQLRequest, SEQUENCE
(snum, atoi (av1[i])));
                                }
                else
                        for (i = 1; i <= QUERIES_PER_SET; i++)
                                {
                                        char qname[10];
                                        sprintf (qname, "%d", SEQUENCE
(snum, i));
                                        qsub (qname, flags);
                                        Run_Query (SQLRequest, SEQUENCE
(snum, i));
                                }
        else if (optind < ac1)
                for (i = optind; i < ac1; i++)
                        {
                                qsub (av1[i], flags);
                                Run_Query (SQLRequest, atoi (av1[i]));
                        }
        else
                for (i = 1; i <= QUERIES_PER_SET; i++)
                        {
```

```
                        char qname[10];
                        sprintf (qname, "%d", i);
                        qsub (qname, flags);
                        Run_Query (SQLRequest, i);
                }

        ftime (&StartTime);
        fprintf (stdout, " Finished query run for stream %d on %s\n",
snum, ctime (&StartTime.time));
        fprintf (fdetail, " Finished query run for stream %d on %s\n",
snum, ctime (&StartTime.time));

        return (0);
}

void
Print_Stream_Summary ()
{
        int i;

        fprintf (fsummary, "\nQuery timing results:\n\n");
        for (i = 1; i <= QUERIES_PER_SET; i++)
                if (QueryTime[i] != 0.0)
                        fprintf (fsummary, "Query #%d time was %.2f
seconds.\n", i, QueryTime[i]);
                else
                        fprintf (fsummary, "Query #%d was not run.\n",
i);

        if (UF1time != 0.0)
                fprintf (fsummary, "UF1 (insert) time was %.2f
seconds.\n", UF1time);
        else
                fprintf (fsummary, "UF1 (insert) was not run.\n");

        if (UF2time != 0.0)
                fprintf (fsummary, "UF2 (delete) time was %.2f
seconds.\n", UF2time);
        else
                fprintf (fsummary, "UF2 (delete) was not run.\n");
}

void
Run_UF1_Inserts ()
{
        ftime (&StartTime);
        fprintf (stdout, " Starting UF1 insert run for set %d on %s\n",
upd_num, ctime (&StartTime.time));
        fprintf (fdetail, "\n Starting UF1 insert run for set %d on
%s\n", upd_num, ctime (&StartTime.time));

        rowcnt = tdefs[ORDER_LINE].base / 10000 * scale * refresh;

        if (rowcnt != 1500 * scale && flt_scale >= 1.0)
                fprintf (stderr, " Why is rowcnt = %d\n", rowcnt);

        minrow = rowcnt * upd_num + 1;
        if (flags & VERBOSE)
                fprintf (fdetail, " Inserting from %d for %d txns,
upd_num =%d\n", minrow, rowcnt, upd_num);

        if (use_fastload_for_updates & 1 == 1)
                gen_updates_via_fastload (minrow, rowcnt, upd_num + 1);
        else
                gen_updates (minrow, rowcnt, upd_num + 1);

        upd_num++;

}

void
Run_UF2_Deletes (int special)
{
        ftime (&StartTime);
        if (special == 0)
                {
                        fprintf (stdout, " Starting UF2 delete run for
set %d on %s\n", upd_num - 1, ctime (&StartTime.time));
                        fprintf (fdetail, "\n Starting UF2 delete run
for set %d on %s\n", upd_num - 1, ctime (&StartTime.time));
                }

        rowcnt = tdefs[ORDER_LINE].base / 10000 * scale * refresh;
        if (rowcnt != 1500 * scale && flt_scale >= 1.0)
                fprintf (stderr, " Why is rowcnt = %d\n", rowcnt);

        minrow = rowcnt * (upd_num - 1) + 1;
        if (flags & VERBOSE)
                fprintf (fdetail, " Deleting from %d for %d txns,
upd_num =%d\n", minrow, rowcnt, upd_num);

        if (use_fastload_for_updates & 2 == 2)
                gen_deletes_via_fastload (minrow, rowcnt, upd_num,
special);
        else
                gen_deletes (minrow, rowcnt, upd_num, special);


}


int
main (int ac, char **av)
{
        int i;
        int c;
        int status;
        FILE *ifp;
        char line[LINE_SIZE];
        double QppD;
        double QthD;
        double QphD;
        double sumoflogs;
        double maxtime;
        double mintime;
        struct timeb ThroughputStartTime;
```

```
        struct timeb ThroughputEndTime;
        struct timeb difTime;
        char tempstr[200];
        char tempstr2[20];
        char tempstr3[128];

        prog = av[0];
        rowcnt = 0;
        table = (1 << CUST) |
                (1 << SUPP) |
                (1 << NATION) |
                (1 << REGION) |
                (1 << PART_PSUPP) |
                (1 << ORDER_LINE);
        scale = 1;
        flt_scale = (float) 1.0;
        flags = 0;
        updates = 1;
        refresh = UPD_PCT;
        gen_sql = 0;
        gen_rng = 1;
        direct = 1;
        tdefs[ORDER].base *=
                ORDERS_PER_CUST;              /* have to do this after
init */
        tdefs[LINE].base *=
                ORDERS_PER_CUST;              /* have to do this after
init */
        tdefs[ORDER_LINE].base *=
                ORDERS_PER_CUST;              /* have to do this after
init */
        children = 1;
        throughputstreams = 3;
        upd_num = 0;
        QppD = 0.0;
        QthD = 0.0;
        QphD = 0.0;

        ac1 = ac;
        av1 = av;


        process_options (ac, av);
        if (flags & VERBOSE)
                printf ("TPC-D Benchmark Driver, (matched to DBGEN/QGEN
Version %d.%d.%d%c)\n",
             VERSION, RELEASE ,MODIFICATION,PATCH);
        fflush (stdout);

        if (use_fastload_for_updates == 3)
                maxSess = throughputstreams + 1;

        if ((flags & SEED) == 0)
                if (flags & EVERYTHING)
                        num_seeds = throughputstreams + 1;

        if (num_seeds > 1 && num_seeds < throughputstreams + 1 &&
(flags & EVERYTHING))
                {
                        fprintf (stderr, " You didn't supply enough seed
values \n");
                        exit (1);
                }
        if (num_seeds > 1 && !(flags & EVERYTHING))
                {
                        fprintf (stderr, " Supplying multiple seeds only
makes sense if you are also use -e \n");
                        exit (1);
                }

        setup ();
        /* have to do this after init */
        tdefs[NATION].base = nations.count;
        tdefs[REGION].base = regions.count;

        for (i = 0; i <= QUERIES_PER_SET; i++)
                SaveSeed[i] = Seed[i];

        if (!(flags & DFLT))        /* preturb the RNG */
            {
            if (!(flags & SEED))
                        {
                        for (i = MAX_STREAM; i >= 0; i--)
                                        {
                                                rndm = rndm =
(long)((unsigned)time(NULL) * DSS_PROC) + i;
                                                if (rndm < 0)
                                                        rndm +=
2147483647;
                                                if (rndm == 0)
                                                        rndm = 12345;
                                                initSeeds[i] = rndm;
                                        }
                    }
                if (num_seeds > 1 && (flags & EVERYTHING))
                                        {
                                                for (i = 0; i < num_seeds;
i++)
                                                        {
                                                                printf (" -
- Using %ld as seed for stream %d\n", initSeeds[i], i);
                                                                if
(initSeeds[i] == 0)
                                                                        {
        fprintf (stderr, " -- Seeds cannot be zero!\n");
        exit (1);
                                                                        }
                                                        }
                                        }
                else
                printf("-- using %ld as a seed to the RNG\n", rndm);

                for (i=1; i <= QUERIES_PER_SET; i++)
                        {
```

```
                    Seed[i] = rndm;
                    UnifInt(0L, 100L, i);
                    UnifInt(0L, 100L, i);
                    UnifInt(0L, 100L, i);
                    UnifInt(0L, 100L, i);
                    UnifInt(0L, 100L, i);
                   }


               }
    else
        printf("-- using default substitutions\n");



        if (!(flags & DBASE))
                {
                        fprintf (stderr, " You must use the -n option to
enter the Teradata logon to use\n");

                        exit (1);
                }


        if (!(flags & OUTPUT))
                {
                        osuff = malloc (strlen (".") + 1);
                        MALLOC_CHECK (osuff);
                        strcpy (osuff, ".");
                }


        if (snum >= 0)
                sprintf (tempstr,
"%s/tpcd_stream_%d_detailed_output.txt", osuff, snum);
        else
                sprintf (tempstr, "%s/tpcd_detailed_output.txt", osuff);
        fdetail = fopen (tempstr, "w");
        if (snum >= 0)
                sprintf (tempstr,
"%s/tpcd_stream_%d_summary_output.txt", osuff, snum);
        else
                sprintf (tempstr, "%s/tpcd_summary_output.txt", osuff);
        fsummary = fopen (tempstr, "w");

        if (fdetail == NULL || fsummary == NULL)
                {
                        fprintf (stderr, " **** Fatal error, cannot open
output files. errno=%d\n", errno);
                        if (flags & OUTPUT)
                                fprintf (stderr, " **** Probably
directory %s does not exist or is not writable\n", osuff);
                        exit (1);
                }

        signal (SIGABRT, stop_proc);
        signal (SIGTERM, stop_proc);
        signal (SIGINT, stop_proc);
#ifdef _WIN32
        signal (SIGBREAK, stop_proc);
#else
        signal (SIGTSTP, stop_proc);
#endif


        fprintf (fdetail, "\n TPC-D Benchmark Execution, detailed
results\n");
        fprintf (fsummary, "\n TPC-D Benchmark Execution, summary
results\n");

        fprintf (fdetail, "\n Using Teradata logon '%s'\n", db_name);
        fprintf (fsummary, "\n Using Teradata logon '%s'\n", db_name);

        fprintf (fsummary, " Using %ld as seed \n", rndm);
        fprintf (fdetail, " Using %ld as seed \n", rndm);

        fprintf (fsummary, " Scale factor is %g\n", flt_scale);
        fprintf (fdetail, " Scale factor is %g\n", flt_scale);

        /* Log on all sessions needed */
        Connect_Sessions ();

        /* Test to see the size of the tables */
        if (specialtest == 0 && snum <= 0)
                Verify_Database ();

        for (i = 0; i <= QUERIES_PER_SET; i++)
                {
                        QueryTime[i] = 0.0;
                }
        UF1time = 0.0;
        UF2time = 0.0;
        /* Run Power test warm-up, optional */
        /* run update function UF1 once on same query stream! */
        if (updates > 0)
                {

                        /* Save Query Seeds */
                        for (i = 0; i <= QUERIES_PER_SET; i++)
                                {
                                        tempSeed = SaveSeed[i];
                                        SaveSeed[i] = Seed[i];
                                        Seed[i] = tempSeed;
                                }



                        if (load_state (scale, children, children))
                                {
                                        fprintf (stderr, "Unable to load
seeds (%d scale)\n",
                                                                scale);
                                        fprintf (stderr, "Either you need
to use the -C option to match\n");
                                        fprintf (stderr, "the value used
in DBGEN, or DBGEN was not run\n");
```

```
                                        fprintf (stderr, "Run './DBGEN -O
s -s %d' and then rerun this program\n", scale);
                                        exit (-1);
                                }


                        if (specialtest)
                                {
                                        upd_num = 0;
                                        system ("prfld");
                                        system ("prfstat");

                                        for (i = 1; i < 5; i++)
                                                {
                                                        setsPerReq = i;
                                                        fprintf (fdetail,
" Testing with setsPerReq = %d \n", setsPerReq);

                                                        upd_num = 1;
                                                        Run_UF2_Deletes
(1);
                                                        fflush (stdout);
                                                        fflush (stderr);
                                                        fflush (fdetail);
                                                        fflush (fsummary);


                                                        sprintf (tempstr,
"prfsnap InsertText%dtxns.log", setsPerReq);
                                                        system (tempstr);
                                                        upd_num = 0;
                                                        Run_UF1_Inserts
();
                                                        system (tempstr);
                                                        fflush (stdout);
                                                        fflush (stderr);
                                                        fflush (fdetail);
                                                        fflush (fsummary);

                                                }
                                        fflush (stdout);
                                        fflush (stderr);
                                        fclose (fsummary);
                                        fclose (fdetail);

                                        exit (1);
                                }

                        if ((flags & DFLT) || snum < 0 || flt_scale <
1.0)
                                {
                                        /* Temp:  Pre-delete to clean up
everything from previous failed runs */
                                        /* normally, this should delete
zero rows */
                                        upd_num = updates;
                                        Run_UF2_Deletes (1);
                                        UF2time = 0.0;
                                }
                                        upd_num = updates - 1;
                                        fflush (fdetail);
                                        fflush (fsummary);

                                        Run_UF1_Inserts ();


                                        /* Restore Query Seeds, Save Insert seeds */
                                        for (i = 0; i <= QUERIES_PER_SET; i++)
                                                {
                                                        tempSeed = SaveSeed[i];
                                                        SaveSeed[i] = Seed[i];
                                                        Seed[i] = tempSeed;
                                                }
                                }

                        fflush (fdetail);
                        fflush (fsummary);

                        if (flags & INIT)                       /* init stream with ifile
*/
                                {
                                        ifp = fopen (ifile, "r");
                                        if (ifp == NULL)
                                                {
                                                        fprintf (stderr, "Failed to open
file '%s'\n", ifile);
                                                        exit (1);
                                                }
                                        while (fgets (line, LINE_SIZE, ifp) != NULL)
                                                fprintf (stdout, "%s", line);
                                }


                        /* Run Power test, stream=0! */

                        Run_Query_Stream ();

                        /* Restore Insert seeds */
                        for (i = 0; i <= QUERIES_PER_SET; i++)
                                {
                                        tempSeed = SaveSeed[i];
                                        SaveSeed[i] = Seed[i];
                                        Seed[i] = tempSeed;
                                }

                        fflush (fdetail);
                        fflush (fsummary);

                        /* Run update function UF2 once on the same query stream! */

                        if (updates > 0)
                                if (snum < 0 || (flags & DFLT))
                                        Run_UF2_Deletes (1);
                                else
                                        Run_UF2_Deletes (0);
```

```
        fprintf (stdout, " Stream %d finished on %s\n", snum, ctime
(&StartTime.time));
        fprintf (fdetail, "\n Stream %d finished on %s\n", snum, ctime
(&StartTime.time));

        Print_Stream_Summary ();

        fflush (fdetail);
        fflush (fsummary);

        if (snum > 0)
                if (flags & EVERYTHING)
                        {
                                Disconnect_Sessions ();

                                fclose (fsummary);
                                fclose (fdetail);

                                return (0);
                        }

        maxtime = 0.0;
        mintime = 9e50;
        for (i = 1; i <= QUERIES_PER_SET; i++)
                {
                        if (maxtime < QueryTime[i])
                                maxtime = QueryTime[i];
                        if (mintime > QueryTime[i])
                                mintime = QueryTime[i];
                }
        if (maxtime > 0.0)
                if ((maxtime / mintime) > 1000.0)
                        {
                                fprintf (fsummary, " Ratio of max query
time to min query time over 1000, special processing in effect\n");
                                fprintf (fdetail, " Ratio of max query
time to min query time over 1000, special processing in effect\n");
                                for (i = 1; i < QUERIES_PER_SET; i++)
                                        if (QueryTime[i] < maxtime /
1000.0)
                                                QueryTime[i] = maxtime /
1000.0;
                        }

        if (maxtime > 0.0)
                {
                        sumoflogs = 0.0;
                        for (i = 1; i <= QUERIES_PER_SET; i++)
                                sumoflogs += log (QueryTime[i]);
                        sumoflogs += log (UF1time) + log (UF2time);
                        QppD = (3600.0 / exp (sumoflogs / 19.0)) *
flt_scale;

                        fprintf (fsummary, "\n QppD@%.1fGB = %.2f\n",
flt_scale, QppD);
                        fprintf (fdetail, "\n QppD@%.1fGB = %.2f\n",
flt_scale, QppD);
                }
```

```
        /* Power test complete!  */


        if (snum == 0)
                if (flags & EVERYTHING)
                        {
                                /*note: nothing is legal between power
test and throughput test */
                                /* Run throughput in parallel with a
single update stream */
                                /*note: stream 1 to s, where s= #streams
*/

                                ftime (&StartTime);
                                ThroughputStartTime.time =
StartTime.time;
                                ThroughputStartTime.millitm =
StartTime.millitm;
                                fprintf (stdout, " Beginning throughput
test using %d query streams on %s\n", throughputstreams, ctime
(&StartTime.time));
                                fprintf (fdetail, "\n Beginning
throughput test using %d query streams on %s\n", throughputstreams,
ctime (&StartTime.time));

                                for (c = 0; c < throughputstreams; c++)
                                        {
                                                sprintf (tempstr, "%d", c
+ 1);
                                                sprintf (tempstr2, "%f",
flt_scale);
                                                if (c < num_seeds)
                                                        sprintf (tempstr3,
"%ld", initSeeds[c + 1]);
                                                else
                                                        sprintf (tempstr3,
"%ld", rndm);


#if (defined(WIN32)&&!defined(_POSIX_))
                                                pids[c] = _spawnl
(_P_NOWAIT, av[0], av[0], "-p", tempstr, "-s", tempstr2, "-U", "0", "-
r", tempstr3, "-n", db_name,

                                        "-o", osuff,

                                        "-v",

                                        (char *) NULL);
                                                fprintf (stdout, " Spawned
task, pid is %d\n", pids[c]);
                                                if (pids[c] == -1)
                                                        {
                                                                fprintf
(stderr, "Child query stream not created %d\n", errno);
                                                                exit (-1);
                                                        }
```

```
#else
                                        pids[c] = SPAWN ();
                                        if (pids[c] == -1)
                                                {
                                                        fprintf
(stderr, "Child query stream not created");
                                                        exit (-1);
                                                }
                                        else if (pids[c] == 0) /*
CHILD */
                                                {
                                                        execl
(av[0], av[0], "-p", tempstr, "-s", tempstr2, "-U", "0", "-r",
tempstr3, "-n", db_name,

"-o", osuff,

(char *) NULL);
                                                        fprintf
(stderr, " Could not execl to start the new task %d\n", errno);

                                                        exit (1);
                                                }
                                        else
#endif
                                        if (flags & VERBOSE)   /*
PARENT */
                                                fprintf (stdout,
".");
                                }

                        if (flags & VERBOSE)
                                fprintf (stdout, "waiting...");

                        fflush (stderr);
                        fflush (fdetail);
                        fflush (fsummary);

                        c = throughputstreams;
                        while (c)
                                {
#if (defined(WIN32)&&!defined(_POSIX_))
                                        i = _cwait (&status,
pids[c - 1], _WAIT_CHILD);
                                        if (i == -1 &&
throughputstreams)
                                                {
                                                        if (errno
== ECHILD)

        fprintf (stderr, "Could not wait on pid %d\n", pids[c - 1]);
                                                        else if
(errno == EINTR)

        fprintf (stderr, "Process %d stopped abnormally\n", pids[c -
1]);
                                                        else if
(errno == EINVAL)

        fprintf (stderr, "Program bug\n");
                                                }
                                        else
                                                {
                                                        fprintf
(stdout, "Process %d: STOPPED\n", pids[c - 1]);
                                                }
#else
                                        i = wait (&status);
                                        if (i == -1 &&
throughputstreams)
                                                {
                                                        fprintf
(stderr, "We lost one\n");
                                                        exit (-2);
                                                }
                                        if (status & 0xFF)
                                                {
                                                        if (status
& 0xFF == 0117)

        printf ("Process %d: STOPPED\n", i);
                                                        else

        printf ("Process %d: rcvd signal %d\n",

                i, status & 0x7F);
                                                }
#endif
                                        c--;
                                }

                        ftime (&StartTime);
                        fprintf (stdout, " Throughput query
streams ended on %s, starting updates\n", ctime (&StartTime.time));
                        fprintf (fdetail, "\n  Throughput query
streams ended on %s, starting updates\n", ctime (&StartTime.time));

                        if (updates > 0)
                                for (c = 0; c <
throughputstreams; c++)
                                        {
                                                Run_UF1_Inserts
();
                                                if (snum < 0 ||
(flags & DFLT))

        Run_UF2_Deletes (1);
                                                else

        Run_UF2_Deletes (0);
                                        }

                        ftime (&ThroughputEndTime);

                        difTime.time = ThroughputEndTime.time -
ThroughputStartTime.time;
```

```
                        if (ThroughputEndTime.millitm >=
ThroughputStartTime.millitm)
                                {
                                        difTime.millitm =
(unsigned short) (ThroughputEndTime.millitm -
ThroughputStartTime.millitm);
                                }
                        else
                                {
                                        difTime.millitm =
(unsigned short) ((1000 + ThroughputEndTime.millitm) -
ThroughputStartTime.millitm);
                                        difTime.time -= 1;
                                };
                        Throughputtime = difTime.time +
difTime.millitm / 1000.0 + 0.005;

                        fprintf (stdout, " Throughput test ended
on %s\n", ctime (&ThroughputEndTime.time));
                        fprintf (fdetail, "\n  Throughput test
ended on %s\n", ctime (&ThroughputEndTime.time));

                        fprintf (fdetail, "\n  Throughput test
took %.2f seconds\n", Throughputtime);

                        if (updates > 0)
                                {
                                        QthD = ((throughputstreams
* 17.0 * 3600.0) / Throughputtime) * flt_scale;

                                        fprintf (fsummary, "\n
QthD@%.1fGB = %.2f\n", flt_scale, QthD);
                                        fprintf (fdetail, "\n
QthD@%.1fGB = %.2f\n", flt_scale, QthD);

                                        QphD = sqrt (QppD * QthD);

                                        fprintf (fsummary, "\n
QphD@%.1fGB = %.2f\n", flt_scale, QphD);
                                        fprintf (fdetail, "\n
QphD@%.1fGB = %.2f\n", flt_scale, QphD);
                                }


                }
```

```
        if (flags & TERMINATE)          /* terminate stream with tfile */
                {
                        ifp = fopen (tfile, "r");
                        if (ifp == NULL)
                                {
                                        fprintf (stderr, "Failed to open
terminate file '%s'\n",
                                                        tfile);
                                        exit (1);
                                }
                        while (fgets (line, LINE_SIZE, ifp) != NULL)
                                fprintf (stdout, "%s", line);
                }

        fflush (fdetail);
        fflush (fsummary);

        Disconnect_Sessions ();

        fclose (fsummary);
        fclose (fdetail);

        return (0);
}
```

vsub.c

```
       /* Sccsid:     @(#)varsub.c   9.1.1.25     5/1/96  11:52:23 */
#include <stdio.h>
#ifndef _POSIX_SOURCE
#include <malloc.h>
#endif /* POSIX_SOURCE */
#if (defined(_POSIX_)||!defined(WIN32))
#include <unistd.h>
#endif /* WIN32 */
#include <string.h>
#include "config.h"
#include "dss.h"
#include "tpcd.h"
#define TYPE_CNT        8
extern long Seed[];
extern char **asc_date;
extern double flt_scale;

void vsub PROTO((char * SQLRequest, int qnum, int vnum, int flags));

char *defaults[19][11] =
    {
    {"90",              NULL,                   NULL,
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 1 */
    {"15",              "BRASS",                "EUROPE",
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 2 */
    {"BUILDING",        "1995-03-15",           NULL,
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 3 */
    {"1993-07-01",      NULL,                   NULL,
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 4 */
    {"ASIA",            "1994-01-01",           NULL,
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 5 */
    {"1994-01-01",      ".06",                  "24",
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 6 */
    {"FRANCE",          "GERMANY",              NULL,
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 7 */
    {"BRAZIL",          "AMERICA",      "ECONOMY ANODIZED STEEL",
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},/* 8 */
    {"green",         NULL,                   NULL,
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 9 */
    {"1993-10-01",      NULL,                   NULL,
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 10 */
    {"GERMANY",         "0.001",                NULL,
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 11 */
    {"MAIL",            "SHIP",                 "1994-01-01",
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 12 */
    {"Clerk#000000088", NULL,                   NULL,
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 13 */
    {"1995-09-01",      NULL,                   NULL,
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 14 */
    {"1996-01-01",      NULL,                   NULL,
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 15 */
    {"Brand#45",        "MEDIUM POLISHED", "49",
                "14","23","45","19","3","36","9", NULL}, /* 16 */
    {"Brand#23",        "MED BOX",              NULL,
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* 17 */
    {NULL,              NULL,                   NULL,
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* UF1 */
    {NULL,              NULL,                   NULL,
        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /* UF2 */
    };
void
vsub(char * SQLRequest, int qnum, int vnum, int flags)
{
    static char param[11][128];
    static FILE *lfp = NULL;
       char tempStr[128];
    char *ptr;
    int i = 0,
        size[10],
        tmp_date;

    if (vnum == 0)
    {
    if ((flags & DFLT) == 0)
    {
    switch(qnum)
        {
        case 1:
            sprintf(param[1], "%d",
UnifInt((long)60,(long)120,(long)qnum));
            param[2][0] = '\0';
            break;
        case 2:
            sprintf(param[1], "%d",
                UnifInt((long)P_SIZE_MIN, (long)P_SIZE_MAX, qnum));
            pick_str(&p_types_set, qnum, param[3]);
            ptr = param[3] + strlen(param[3]);
            while (*(ptr - 1) != ' ') ptr--;
            strcpy(param[2], ptr);
            pick_str(&regions, qnum, param[3]);
            param[4][0] = '\0';
            break;
        case 3:
            pick_str(&c_mseg_set, qnum, param[1]);
            /*
             * pick a random offset within the month of march and add
the
             * appropriate magic numbers to position the output
functions
             * at the start of March '95
             */
            tmp_date = UnifInt((long)0, (long)30, (long)qnum);
           strcpy(param[2], *(asc_date + tmp_date + 1155));
            if (oldtime)
                {
                for (i=0;strcmp(*(asc_date + i), param[2]); i++);
```

```
            sprintf(param[2],"%ld", julian(i + STARTDATE));
            }
        param[3][0] = '\0';
        break;
    case 4:
        tmp_date = UnifInt(1,58,qnum);
        sprintf(param[1],"19%02d-%02d-01",
            93 + tmp_date/12, tmp_date%12 + 1);
        if (oldtime)
            {
            for (i=0;strcmp(*(asc_date + i), param[1]); i++);
            sprintf(param[1],"%ld", julian(i + STARTDATE));
            }
        param[2][0] = '\0';
        break;
    case 5:
        pick_str(&regions, qnum, param[1]);
        tmp_date = UnifInt((long)93,(long)97,(long)qnum);
        if (oldtime) sprintf(param[2],"%ld", tmp_date);
        else sprintf(param[2], "19%d-01-01", tmp_date);
        param[3][0] = '\0';
        break;
    case 6:
        tmp_date = UnifInt(93,97,qnum);
        if (oldtime)
            sprintf(param[1],"%ld001", tmp_date);
        else
            sprintf(param[1], "19%d-01-01", tmp_date);
        sprintf(param[2], "0.0%d", UnifInt(2, 9, qnum));
        sprintf(param[3], "%d", UnifInt((long)24, (long)25,
(long)qnum));
        param[4][0] = '\0';
        break;
    case 7:
        tmp_date = pick_str(&nations2, qnum, param[1]);
        while (pick_str(&nations2, qnum, param[2]) == tmp_date);
        param[3][0] = '\0';
        break;
    case 8:
        tmp_date = pick_str(&nations2, qnum, param[1]);
        tmp_date = nations.list[tmp_date].weight;
        strcpy(param[2], regions.list[tmp_date].text);
        pick_str(&p_types_set, qnum, param[3]);
        param[4][0] = '\0';
        break;
    case 9:
        pick_str(&colors, qnum, param[1]);
        param[2][0] = '\0';
        break;
    case 10:
        tmp_date = UnifInt(1,24,qnum);
        sprintf(param[1],"19%02d-%02d-01",
            93 + tmp_date/12, tmp_date%12 + 1);
        if (oldtime)
            {
            for (i=0;strcmp(*(asc_date + i), param[1]); i++);
            sprintf(param[1],"%ld", julian(i + STARTDATE));
            }
        param[2][0] = '\0';
        break;
    case 11:
        pick_str(&nations2, qnum, param[1]);
        sprintf(param[2], "%11.10f", Q11_FRACTION / flt_scale );
        param[3][0] = '\0';
        break;
    case 12:
        tmp_date = pick_str(&l_smode_set, qnum, param[1]);
        while (tmp_date == pick_str(&l_smode_set, qnum, param[2]));
        tmp_date = UnifInt(93,97,qnum);
        if (oldtime) sprintf(param[3],"%d", tmp_date*1000 + 1);
        else sprintf(param[3], "19%d-01-01", tmp_date);
        param[4][0] = '\0';
        break;
    case 13:
        sprintf(param[1], O_CLRK_FMT, O_CLRK_TAG,
            UnifInt((long)1,
                   (long) MAX((scale * O_CLRK_SCL), O_CLRK_SCL),
                   (long)qnum));

        param[2][0] = '\0';
        break;
    case 14:
        tmp_date = UnifInt(1,60,qnum);
        sprintf(param[1],"19%02d-%02d-01",
            93 + tmp_date/12, tmp_date%12 + 1);
        if (oldtime)
            {
            for (i=0;strcmp(*(asc_date + i), param[1]); i++);
            sprintf(param[1],"%ld", julian(i + STARTDATE));
            }
        param[2][0] = '\0';
        break;
    case 15:
        tmp_date = UnifInt(1,58,qnum);
        sprintf(param[1],"19%02d-%02d-01",
            93 + tmp_date/12, tmp_date%12 + 1);
        if (oldtime)
            {
            for (i=0;strcmp(*(asc_date + i), param[1]); i++);
            sprintf(param[1],"%ld", julian(i + STARTDATE));
            }
        param[2][0] = '\0';
        break;
    case 16:
        sprintf(param[1], "Brand#%d%d",
            UnifInt(1, 5, qnum),
            UnifInt(1, 5, qnum));
        pick_str(&p_types_set, qnum, param[2]);
        ptr = param[2] + strlen(param[2]);
        while (*(--ptr) != ' ');
        *ptr = '\0';
        i=0;
next:
        size[i] = UnifInt(1, 50, qnum);
        tmp_date = 0;
        while (tmp_date < i)
```

```
            if (size[i] == size[tmp_date])
                goto next;
            else
                {
                tmp_date++;
                sprintf(param[i + 2], "%d", size[i]);
                }
        if (++i <= TYPE_CNT)
            goto next;
        param[i + 2][0] = '\0';
        break;
    case 17:
        sprintf(param[1], "Brand#%ld%ld",
            UnifInt(1, 5, qnum),
            UnifInt(1, 5, qnum));
        pick_str(&p_cntr_set, qnum, param[2]);
        param[3][0] = '\0';
        break;
    case 18:
    case 19:
            break;
    default:
        fprintf(stderr,
            "No variable definitions available for query %d\n",
                qnum);
    }
}

if (flags & LOG)
    {
    if (lfp == NULL)
        {
        lfp = fopen(lfile, "a");
        if (lfp == NULL)
            {
            fprintf(stderr, "Failed to open log file '%s'\n",
                lfile);
            exit(1);
            }
        }
    fprintf(lfp, "%d", qnum);
    for (i=1; i <= 10; i++)
        if (flags & DFLT)
           {
           if (defaults[i - 1] == NULL)
               break;
           else
               fprintf(lfp, "\t%s", defaults[i - 1]);
           }
        else
           {
           if (param[i][0] == '\0')
               break;
           else
               fprintf(lfp, "\t%s", param[i]);
           }
    fprintf(lfp, "\n");
    }
    }
    else
        {
        if (flags & DFLT)
            {
            /* to allow -d to work at all scale factors */
            if (qnum == 11 && vnum == 2)
                                {
                sprintf(tempStr, "%11.10f", Q11_FRACTION/flt_scale);
                                strcat(SQLRequest,tempStr);
                                }
            else
                if (defaults[qnum - 1][vnum - 1])
                    strcat(SQLRequest, defaults[qnum - 1][vnum - 1]);
                else
                   fprintf(stderr,
                       "Bad default request (q: %d, p: %d)\n",
                        qnum, vnum);
            }
        else
            {
            if (param[vnum])
                strcat(SQLRequest,  param[vnum]);
            else
               fprintf(stderr, "Bad parameter request (q: %d, p:
%d)\n",
                    qnum, vnum);
            }
        }
    return;
}
```

Implementation specific layer

To simplify the source to the tpcddriver, the details of calling Teradata CLIv2 was hidden in this layer of code designed to look similar to ISO CLI or ODBC CLI.

Tdatsql.h

```
/*
 * tdatsql.h This file just includes the standard ODBC or ISO CLI
 * header files (see ISO/IEC 9075-3:1995, or the Microsoft ODBC SDK
 *
 */

#undef FAR
#define FAR

typedef void * HWND;
#include <sqltypes.h>
#include <sql.h>


/* Two Teradata-specific functions, because this made my life
   easier than using the standard ISO CLI or ODBC function */


SQLRETURN SQL_API  SQLSetUsingParcel(
       SQLHSTMT  hstmt,
    char * uptr,
    long ulen);

SQLRETURN SQL_API  SQLWaitForAny(
       SQLHENV   henv,
       SQLHSTMT  * lphstmt,
    int * stmtno);
```

tsqlt.h

```
/*
 * tsqlt.h
 * Teradata SQL Type codes are different from ODBC SQL Type codes,
 * so we need to translate them.  This translates from Teradata form
 * to ODBC form.   The Teradata form also contains the Nullable
information,
 * so we extract that at the same time.
 */

void TranslateSQLType( SWORD * SqlType, SWORD * Nullable, SWORD
OldSqlType );


/* This translates the other way, from ODBC SQL Type to Teradata SQL
Type.   */
#ifdef __cplusplus
extern "C"
#endif
SWORD UnTranslateSQLType( SWORD SqlType );
```

tsqlt.c

```
/*
 * TSQLT.C
 * Teradata SQL Type codes are different from ODBC SQL Type codes,
 * so we need to translate them.  This translates from Teradata form
 * to ODBC form.   The Teradata form also contains the Nullable
information,
 * so we extract that at the same time.
 */
#include "tdatsql.h"

void TranslateSQLType( SWORD * SqlType, SWORD * Nullable, SWORD
OldSqlType )
    {
    switch ( OldSqlType )
       {
 case 384:                      // IBM Date
 case 385:
 case 570:                      // DBC date (old style)?
 case 571:
 case 572:
 case 573:
 case 752:                      // DBC date
 case 753:
           *SqlType = SQL_DATE;
           break;

         case 388:
         case 389:
           *SqlType = SQL_TIME;
           break;

         case 392:
         case 393:
           *SqlType = SQL_TIMESTAMP;
           break;

         case 448:
         case 449:
           *SqlType = SQL_VARCHAR;
           break;

         case 452:
         case 453:
           *SqlType = SQL_CHAR;
           break;

         case 456:
         case 457:
           *SqlType = SQL_LONGVARCHAR;
           break;

         case 460:
         case 461:
           *SqlType = SQL_VARCHAR;     // with zero byte at end.
           break;
```

```
case 468:             // GRAPHIC
case 469:
  *SqlType = SQL_CHAR;// ?
  break;

case 464:             // VARGRAPHIC
case 465:
case 472:             // LONG VARGRAPHIC
case 473:
  *SqlType = SQL_VARCHAR;     // ?
  break;

case 480:
case 481:
  *SqlType = SQL_FLOAT;
  break;

case 484:
case 485:
  *SqlType = SQL_DECIMAL;
  break;

case 496:
case 497:
  *SqlType = SQL_INTEGER;
  break;

case 500:
case 501:
  *SqlType = SQL_SMALLINT;
  break;

case 576:
case 577:
case 756:
case 757:
  *SqlType = SQL_TINYINT;
  break;

case 588:
case 589:
case 692:
case 693:
  *SqlType = SQL_BINARY;
  break;

case 582:
case 583:
case 688:
case 689:
case 594:
case 595:
case 696:
case 697:
  *SqlType = SQL_VARBINARY;
  break;
```

```
          default:
               ;
            }
    if ( ( OldSqlType & 1 ) == 1 )
        *Nullable = SQL_NULLABLE;
    else
        *Nullable = SQL_NO_NULLS;
    }

// This translates the other way, from ODBC SQL Type to Teradata SQL
Type.
// SDK 2.0 introduced signed/unsigned LONG, SHORT & TINYINT. Signed
types map
// to the same SqlType as SDK 1.0, but for unsigned they have to map to
a larger
// type. This was added to fix GCA list #1011, where parameter passing
// (binding) of signed/unsigned  types was not working.
SWORD UnTranslateSQLType( SWORD SqlType )
    {
    switch ( SqlType )
        {
 case SQL_DATE:
 case SQL_TIMESTAMP:
            return 753;         // DBC Date
            break;

          case SQL_VARCHAR:
            return 449;
            break;

          case SQL_CHAR:
            return 453;
            break;

          case SQL_LONGVARCHAR:
            return 457;
            break;

          case SQL_REAL:
               return 481;    // Teradata does not have single prec
real.

          case SQL_FLOAT:
          case SQL_DOUBLE:
          case SQL_TIME:
            return 481;

          case SQL_DECIMAL:
          // Rudy Ezquerro. GCA list #1011, [un]signed types not bound
properly
          // SQL_C_ULONG = SQL_INTEGER + SQL_UNSIGENED_OFFSET
          // Best match for ULONG is DECIMAL(10,0) ==> 8 bytes
          case SQL_C_ULONG:
            return 485;

          case SQL_INTEGER:
          // SQL_C_SLONG = SQL_INTEGER + SQL_SIGNED_OFFSET
          // SQL_C_USHORT = SQL_SMALLINT + SQL_UNSIGED_OFFSET
```

```
        case SQL_C_SLONG:
        case SQL_C_USHORT:
          return 497;

        case SQL_SMALLINT:
        // SQL_C_SSHORT = SQL_SMALLINT + SQL_SIGNED_OFFSET
        // SQL_C_UTINYINT = SQL_TINYINT + SQL_UNSIGNED_OFFSET
        case SQL_C_SSHORT:
        case SQL_C_UTINYINT:
          return 501;

        case SQL_TINYINT:
        case SQL_BIT:
        // SQL_C_STINYINT = SQL_TINYINT + SQL_SIGNED_OFFSET
        case SQL_C_STINYINT:
          return 757;

        case SQL_BINARY:
          return 693;

        case SQL_VARBINARY:
          return 689;

        default:
          return SqlType;
          }
    }
```

tdatsql.c

```
/*
 * tdatsql.c  This is an interface layer that gives a very primitive
 * ODBC style (or ISO CLI style) call level interface look on top of
 * Teradata's standard CLIv2.  This allows the tpcddriver.c to be
 * a bit more readable and cleaner, and we hide the CLIv2 stuff in
 * hear.  Only enough of ODBC/ISO CLI is implemented to run just
 * what we need for TPC-D, although this code could be used for other
 * things.
 * The idea is that someday, when we have either an ANSI X/Open
 * or ISO CLI on Unix, or an ODBC on Unix, we can just rip out this
 * code and throw it away, and use the ISO or ODBC cli.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <limits.h>
#include <math.h>
#include <sys/types.h>
#include <sys/timeb.h>
#if (defined(_POSIX_)||!defined(WIN32))
#include <unistd.h>
#define INLINE _Inline
#ifndef snprintf
#define snprintf(w,x,y,z) sprintf(w,y,z)
#endif
#else
#include <process.h>
/*#include <signal.h> */
#include <errno.h>
/*
   #pragma warning(disable:4201)
   #pragma warning(disable:4214)
   #pragma warning(disable:4514)
   #define WIN32_LEAN_AND_MEAN
   #define NOATOM
   #define NOGDICAPMASKS
   #define NOMETAFILE
   #define NOMINMAX
   #define NOMSG
   #define NOOPENFILE
   #define NORASTEROPS
   #define NOSCROLL
   #define NOSOUND
   #define NOSYSMETRICS
   #define NOTEXTMETRIC
   #define NOWH
   #define NOCOMM
   #define NOKANJI
   #define NOMCX
   #include <windows.h>
   #pragma warning(default:4201)
   #pragma warning(default:4214)
 */
#define INLINE _inline
#ifndef itoa
#define itoa _itoa
#endif
#ifndef snprintf
#define snprintf _snprintf
#endif
#endif
#include <ctype.h>
#include <time.h>
#include <assert.h>
#include "tdatsql.h"
#include "tsqlt.h"

#include <coptypes.h>
#include <dbcarea.h>
#include <coperr.h>
#include <dbcerr.h>
#include <parcel.h>

#ifndef __min
#define __min(a,b)     (((a) < (b)) ? (a) : (b))
#endif

#define MALLOC_CHECK(var) \
    if ((var) == NULL) \
        { \
        fprintf(stderr, "Malloc failed at %s:%d\n",  \
            __FILE__, __LINE__); \
```

```
        exit(1);\
        }
#define DBCINTN    496
#define DBCINTI    497
#define DBCSINTN   500
#define DBCSINTI   501
#define DBCDECN    484
#define DBCDECI    485
#define DBCFLOATN  480
#define DBCFLOATI  481
#define DBCVCHARN  448
#define DBCVCHARI  449
#define DBCCHARN   452
#define DBCCHARI   453
#define DBCLCHARN  456
#define DBCLCHARI  457
#define DBCDATEN   752
#define DBCDATEI   753
#define DBCBINTN   756
#define DBCBINTI   757
#define DBCVBYTEN  688
#define DBCVBYTEI  689
#define DBCBYTEN   692
#define DBCBYTEI   693
#define DBCLBYTEN  696
#define DBCLBYTEI  697

#define LINE_SIZE 512



char *CLICTXT = NULL;

long hstmt_next_num = 1;

/*
   INLINE
   void DR_VStr(char * x)
   {
   short len;
   len=(short)strlen(x);
   memcpy(p1,&len,2);
   p1 += 2;
   memcpy(p1,x,len);
   p1 += len;
   }
   INLINE
   void DR_Str(char * x,size_t y)
   {
   memset(p1,' ',y);
   memcpy(p1,x,strlen(x));
   p1+=y;
   }
   INLINE
   void DR_Money(long x)
   {
   long temp = 0;
   memcpy(p1,&x,4);
   p1 += 4;
   memcpy(p1,&temp,4);
   p1 += 4;

   }
   INLINE
   void DR_Char(char x)
   {
   *p1 = x; p1++;
   }
   INLINE
   void DR_Date(char * x)
   {
   long tempdate;
   int y,m,d;
   y=atoi(x);
   m=atoi(x+5);
   d=atoi(x+8);
   if (y<1800 || y>2200 || m<1 || m>12 || d<1 || d>31)
   {
   fprintf(stderr," Help! I can't interpret the date %s\n",x);
   exit(1);
   }
   tempdate=(y-1900)*10000+m*100+d;
   memcpy(p1,&tempdate,4);
   p1 += 4;

   }
 */

typedef struct DBC
       {

               Int32 SessionId;
               Int32 ReqId;
               short numResultCols;
               long RowCount;
               char *p;
               unsigned short Async;
               unsigned short busy;
               long NativeError;
               char ErrorMessage[512];
               struct CliDataInfoType DataInfo;
               struct DBCAREA dbcarea;

       }
DBC;

typedef struct STMT
       {

               DBC *lpdbc;
               long hstmt_num;
       }
STMT;
```

```
SQLRETURN SQL_API
SQLAllocConnect (
                                        SQLHENV henv,
                                        SQLHDBC FAR * phdbc)
{
        Int32 ReturnCode;
        struct DBC *lpdbc;
        lpdbc = malloc (sizeof (struct DBC));
        lpdbc->SessionId = 0;
        lpdbc->ReqId = 0;
        lpdbc->numResultCols = 0;
        lpdbc->RowCount = 0;
        lpdbc->p = NULL;
        lpdbc->Async = 0;
        lpdbc->busy = 0;
        lpdbc->NativeError = (-999);
        lpdbc->ErrorMessage[0] = '\0';
        memset (&lpdbc->DataInfo, 0, sizeof (lpdbc->DataInfo));
        memset (&lpdbc->dbcarea, 0, sizeof (lpdbc->dbcarea));


        strncpy (lpdbc->dbcarea.eyecatcher, "DBCAREA", 8);
        lpdbc->dbcarea.total_len = sizeof (struct DBCAREA);
        DBCHINI (&ReturnCode, CLICTXT, &lpdbc->dbcarea);
        if (ReturnCode != 0)
                {
                        fprintf (stderr, " The DBCHINI return code was
%ld\n", ReturnCode);
                        strncpy (lpdbc->ErrorMessage, lpdbc-
>dbcarea.msg_text, lpdbc->dbcarea.msg_len);
                        lpdbc->ErrorMessage[lpdbc->dbcarea.msg_len] =
'\0';
                        lpdbc->NativeError = ReturnCode;
                        return (SQL_ERROR);
                }

        assert (lpdbc->dbcarea.total_len == sizeof (struct DBCAREA));
        strncpy (lpdbc->dbcarea.eyecatcher, "DBCAREA", 8);

        lpdbc->dbcarea.req_buf_len = 32700;
        lpdbc->dbcarea.resp_buf_len = 32700;
        lpdbc->dbcarea.max_num_sess = 100;
        lpdbc->dbcarea.resp_mode = 'I';
        lpdbc->dbcarea.keep_resp = 'N';
        lpdbc->dbcarea.use_presence_bits = 'N';
        lpdbc->dbcarea.wait_for_resp = 'Y';
        lpdbc->dbcarea.loc_mode = 'Y';
        lpdbc->dbcarea.var_len_req = 'N';
        lpdbc->dbcarea.var_len_fetch = 'N';
        lpdbc->dbcarea.save_resp_buf = 'N';
        lpdbc->dbcarea.two_resp_bufs = 'N';
        lpdbc->dbcarea.req_proc_opt = 'E';
        lpdbc->dbcarea.change_opts = 'Y';
        lpdbc->dbcarea.wait_for_resp = 'Y';
        lpdbc->dbcarea.ret_time = 'Y';
        lpdbc->dbcarea.parcel_mode = 'Y';
        lpdbc->dbcarea.tell_about_crash = 'Y';       /* Set crash
options */
        lpdbc->dbcarea.wait_across_crash = 'N';             /* Set
crash options */
        lpdbc->dbcarea.msg_security = 'N';
        *phdbc = (SQLHDBC) lpdbc;
        return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLAllocEnv (
                                  SQLHENV FAR * phenv)
{
        return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLAllocStmt (
                                   SQLHDBC hdbc,
                                   SQLHSTMT FAR * phstmt)
{
        struct STMT *lpstmt;
        lpstmt = malloc (sizeof (struct STMT));
        lpstmt->lpdbc = (DBC *) hdbc;
        lpstmt->hstmt_num = hstmt_next_num++;
        *phstmt = (SQLHSTMT) lpstmt;
        return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLSetStmtOption (
                                          SQLHSTMT hstmt,
                                          SQLUSMALLINT fOption,
                                          SQLUINTEGER vParam)
{
        if (fOption == SQL_ASYNC_ENABLE)
                ((STMT *) hstmt)->lpdbc->Async = (USHORT) vParam;

        if (((STMT *) hstmt)->lpdbc->Async != SQL_ASYNC_ENABLE_ON)
                ((STMT *) hstmt)->lpdbc->dbcarea.wait_for_resp = 'Y';
        return (SQL_SUCCESS);
}


SQLRETURN SQL_API
SQLColAttributes (
                                          SQLHSTMT hstmt,
                                          SQLUSMALLINT icol,
                                          SQLUSMALLINT fDescType,
                                          SQLPOINTER rgbDesc,
                                          SQLSMALLINT cbDescMax,
                                          SQLSMALLINT FAR * pcbDesc,
                                          SQLINTEGER FAR * pfDesc)
{
        SWORD SType;
        SWORD Nullable;
        char temp[20];
        sprintf (temp, "%hd", icol);
```

```
        if (icol > 0)
                {
                        TranslateSQLType (&SType, &Nullable, ((STMT *)
hstmt)->lpdbc->DataInfo.InfoVar[icol - 1].SQLType);
                }
        switch (fDescType)
                {
                case SQL_COLUMN_LABEL:
                case SQL_COLUMN_NAME:
                        {
                                strcpy (rgbDesc, "Col");
                                strcat (rgbDesc, temp);
                                *pcbDesc = (SQLSMALLINT) strlen ((char *)
rgbDesc);
                                return (SQL_SUCCESS);
                        }
                case SQL_COLUMN_TYPE:
                        *pfDesc = (long) SType;
                        break;

                case SQL_COLUMN_LENGTH:
                        *pfDesc = ((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLLen;
                        switch (SType)
                                {
                                case SQL_BIT:
                                case SQL_TINYINT:
                                        *pfDesc = 1;
                                        break;
                                case SQL_SMALLINT:
                                        *pfDesc = 2;
                                        break;
                                case SQL_INTEGER:
                                        *pfDesc = 4;
                                        break;
                                case SQL_BIGINT:
                                        *pfDesc = 20;
                                        break;
                                case SQL_REAL:
                                        *pfDesc = 4;
                                        break;
                                case SQL_FLOAT:
                                case SQL_DOUBLE:
                                        *pfDesc = 8;
                                        break;
                                case SQL_DATE:
                                case SQL_TIME:
                                        *pfDesc = 6;
                                        break;
                                case SQL_TIMESTAMP:
                                        *pfDesc = 16;
                                        break;
                                case SQL_NUMERIC:
                                case SQL_DECIMAL:
                                        *pfDesc = ((STMT *) hstmt)-
>lpdbc->DataInfo.InfoVar[icol - 1].SQLLen + 2;
                                        break;
                                default:
                                        *pfDesc = ((STMT *) hstmt)-
>lpdbc->DataInfo.InfoVar[icol - 1].SQLLen;
                                }

                        break;
                case SQL_COLUMN_PRECISION:
                        switch (SType)
                                {
                                case SQL_BIT:
                                        *pfDesc = 1;
                                        break;
                                case SQL_TINYINT:
                                        *pfDesc = 3;
                                        break;
                                case SQL_SMALLINT:
                                        *pfDesc = 5;
                                        break;
                                case SQL_INTEGER:
                                        *pfDesc = 10;
                                        break;
                                case SQL_BIGINT:
                                        *pfDesc = 19;
                                        break;
                                case SQL_REAL:
                                        *pfDesc = 7;
                                        break;
                                case SQL_FLOAT:
                                case SQL_DOUBLE:
                                        *pfDesc = 15;
                                        break;
                                case SQL_DATE:
                                        *pfDesc = 10;
                                        break;
                                case SQL_TIME:
                                        *pfDesc = 8;
                                        break;
                                case SQL_TIMESTAMP:
                                        *pfDesc = 23;
                                        break;
                                case SQL_DECIMAL:
                                        *pfDesc = ((STMT *) hstmt)-
>lpdbc->DataInfo.InfoVar[icol - 1].SQLLen / 256;
                                        break;
                                default:
                                        *pfDesc = ((STMT *) hstmt)-
>lpdbc->DataInfo.InfoVar[icol - 1].SQLLen;
                                }
                        break;
                case SQL_COLUMN_SCALE:
                        *pfDesc = 0;
                        if (SType == SQL_DECIMAL)
                                *pfDesc = ((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLLen & 0x00FF;
                        break;
                case SQL_COLUMN_DISPLAY_SIZE:
                        switch (SType)
                                {
                                case SQL_BIT:
```

```
                                *pfDesc = 1;
                                break;
                        case SQL_TINYINT:
                                *pfDesc = 4;
                                break;
                        case SQL_SMALLINT:
                                *pfDesc = 6;
                                break;
                        case SQL_INTEGER:
                                *pfDesc = 11;
                                break;
                        case SQL_BIGINT:
                                *pfDesc = 20;
                                break;
                        case SQL_REAL:
                                *pfDesc = 13;
                                break;
                        case SQL_FLOAT:
                        case SQL_DOUBLE:
                                *pfDesc = 22;
                                break;
                        case SQL_DATE:
                                *pfDesc = 10;
                                break;
                        case SQL_TIME:
                                *pfDesc = 8;
                                break;
                        case SQL_TIMESTAMP:
                                *pfDesc = 23;
                                break;
                        case SQL_BINARY:
                        case SQL_VARBINARY:
                        case SQL_LONGVARBINARY:
                                *pfDesc = ((STMT *) hstmt)-
>lpdbc->DataInfo.InfoVar[icol - 1].SQLLen * 2;
                        case SQL_NUMERIC:
                        case SQL_DECIMAL:
                                *pfDesc = ((STMT *) hstmt)-
>lpdbc->DataInfo.InfoVar[icol - 1].SQLLen / 256 + 2;
                                break;

                        default:
                                *pfDesc = *pfDesc = ((STMT *)
hstmt)->lpdbc->DataInfo.InfoVar[icol - 1].SQLLen;
                        }

                break;

        case SQL_COLUMN_NULLABLE:
                *pfDesc = Nullable;
                break;

        default:
                return (SQL_ERROR);
        }

    return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLRowCount (
                                SQLHSTMT hstmt,
                                SQLINTEGER FAR * pcrow)
{
    *pcrow = ((STMT *) hstmt)->lpdbc->RowCount;
    return (SQL_SUCCESS);
}
SQLRETURN SQL_API
SQLNumResultCols (
                                        SQLHSTMT hstmt,
                                        SQLSMALLINT FAR * pccol)
{
    *pccol = (short) ((STMT *) hstmt)->lpdbc->numResultCols;
    return (SQL_SUCCESS);
}
SQLRETURN SQL_API
SQLError (
                        SQLHENV henv,
                        SQLHDBC hdbc,
                        SQLHSTMT hstmt,
                        SQLCHAR FAR * szSqlState,
                        SQLINTEGER FAR * pfNativeError,
                        SQLCHAR FAR * szErrorMsg,
                        SQLSMALLINT cbErrorMsgMax,
                        SQLSMALLINT FAR * pcbErrorMsg)
{

    DBC *lpdbc;
    if (hstmt != 0)
            lpdbc = ((STMT *) hstmt)->lpdbc;
    else
            lpdbc = (DBC *) hdbc;
    *pfNativeError = lpdbc->NativeError;
    if (lpdbc->NativeError == 0)
            strcpy ((char *) szSqlState, "00000");
    else
            strcpy ((char *) szSqlState, "S1000");
    strcpy ((char *) szErrorMsg, lpdbc->ErrorMessage);
    *pcbErrorMsg = (SQLSMALLINT) strlen (lpdbc->ErrorMessage);
    if (lpdbc->NativeError == 0)
            return (SQL_NO_DATA_FOUND);
    else
            return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLFreeStmt (
                                SQLHSTMT hstmt,
                                SQLUSMALLINT fOption)
{
    Int32 ReturnCode;
    ((STMT *) hstmt)->lpdbc->dbcarea.i_sess_id = ((STMT *) hstmt)-
>lpdbc->SessionId;
    ((STMT *) hstmt)->lpdbc->dbcarea.i_req_id = ((STMT *) hstmt)-
>lpdbc->ReqId;
```

```
        ((STMT *) hstmt)->lpdbc->dbcarea.func = DBFERQ;
        DBCHCL (&ReturnCode, CLICTXT, &((STMT *) hstmt)->lpdbc-
>dbcarea);

        ((STMT *) hstmt)->lpdbc->dbcarea.using_data_len = 0;
        ((STMT *) hstmt)->lpdbc->dbcarea.using_data_ptr = NULL;

        if (ReturnCode != 0)
                {
                        if (ReturnCode != 305) /* Session not found */
                                {
                                        fprintf (stderr, " The ERQ return
code was %ld\n", ReturnCode);
                                        strncpy (((STMT *) hstmt)->lpdbc-
>ErrorMessage, ((STMT *) hstmt)->lpdbc->dbcarea.msg_text, ((STMT *)
hstmt)->lpdbc->dbcarea.msg_len);
                                        ((STMT *) hstmt)->lpdbc-
>ErrorMessage[((STMT *) hstmt)->lpdbc->dbcarea.msg_len] = '\0';
                                        ((STMT *) hstmt)->lpdbc-
>NativeError = ReturnCode;
                                        return (SQL_ERROR);
                                }
                }
        return (SQL_SUCCESS);
}



SQLRETURN SQL_API
SQLGetData (
                        SQLHSTMT hstmt,
                        SQLUSMALLINT icol,
                        SQLSMALLINT fCType,
                        SQLPOINTER rgbValue,
                        SQLINTEGER cbValueMax,
                        SQLINTEGER FAR * pcbValue)
{

        SWORD SType;
        SWORD Nullable;
        char *p2;
        static char tempStr[512];
        double tempFloat;
        unsigned long tempUInt;
        long tempInteger;
        short tempSmallint;


        ((STMT *) hstmt)->lpdbc->NativeError = 0;


        if ((int) icol > (int) ((STMT *) hstmt)->lpdbc->numResultCols)
                return (SQL_NO_DATA_FOUND);
        assert (((STMT *) hstmt)->lpdbc->numResultCols > 0);
        assert (icol > 0 && (int) icol <= (int) ((STMT *) hstmt)-
>lpdbc->numResultCols);
        assert (((STMT *) hstmt)->lpdbc->dbcarea.fet_ret_data_len > 0);
        if (icol == 1)
                {
                        ((STMT *) hstmt)->lpdbc->p = ((struct
CliRecordType *) ((STMT *) hstmt)->lpdbc->dbcarea.fet_data_ptr)->Body;
                        assert (((STMT *) hstmt)->lpdbc->p != NULL);

                        ((STMT *) hstmt)->lpdbc->p += ((((STMT *)
hstmt)->lpdbc->numResultCols + 7) / 8);      /* Skip Indicator bytes */
                }
        p2 = ((struct CliRecordType *) ((STMT *) hstmt)->lpdbc-
>dbcarea.fet_data_ptr)->Body +
                ((STMT *) hstmt)->lpdbc->dbcarea.fet_ret_data_len;

        if (icol > 0)
                {
                        TranslateSQLType (&SType, &Nullable, ((STMT *)
hstmt)->lpdbc->DataInfo.InfoVar[icol - 1].SQLType);
                }
        assert (((STMT *) hstmt)->lpdbc->p != NULL);
        assert (p2 != NULL);
        assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
        assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len == sizeof
(struct DBCAREA));

        if (((STMT *) hstmt)->lpdbc->p >= p2)
                return (SQL_NO_DATA_FOUND);

        if (fCType == SQL_C_DEFAULT &&
                (SType == SQL_CHAR || SType == SQL_VARCHAR))
                fCType = SQL_C_CHAR;

        if (fCType == SQL_C_DEFAULT &&
                (SType == SQL_INTEGER))
                fCType = SQL_C_SLONG;

        assert (fCType == SQL_C_CHAR || fCType == SQL_C_LONG || fCType
== SQL_C_SLONG);

        if (pcbValue != NULL)
                *pcbValue = 0;

        switch (((STMT *) hstmt)->lpdbc->DataInfo.InfoVar[icol -
1].SQLType & 0xFFFE)
                {
                case 496:                               /* integer */
                        memcpy (&tempInteger, ((STMT *) hstmt)->lpdbc-
>p, 4);
                        ((STMT *) hstmt)->lpdbc->p += 4;
                        break;
                case 500:                               /* smallint */
                        memcpy (&tempSmallint, ((STMT *) hstmt)->lpdbc-
>p, 4);
                        tempInteger = tempSmallint;
                        ((STMT *) hstmt)->lpdbc->p += 2;
                        break;
                case 484:                               /* decimal */
```

```
                        if (((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLLen / 256 > 9)
                                {

                                        memcpy (&tempUInt, ((STMT *)
hstmt)->lpdbc->p, 4);
                                        tempFloat = tempUInt;

                                        memcpy (&tempInteger, ((STMT *)
hstmt)->lpdbc->p + 4, 4);
                                        tempFloat += tempInteger *
4294967296.0;

                                        ((STMT *) hstmt)->lpdbc->p += 8;
                                }
                        else
                                {
                                        if (((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLLen / 256 > 4)
                                                {
                                                        memcpy
(&tempInteger, ((STMT *) hstmt)->lpdbc->p, 4);
                                                        tempFloat =
tempInteger;
                                                        ((STMT *) hstmt)-
>lpdbc->p += 4;
                                                }
                                        else
                                                {
                                                        if (((STMT *)
hstmt)->lpdbc->DataInfo.InfoVar[icol - 1].SQLLen / 256 > 2)
                                                                {

        memcpy (&tempSmallint, ((STMT *) hstmt)->lpdbc->p, 2);

        tempInteger = tempSmallint;

        tempFloat = tempSmallint;

        ((STMT *) hstmt)->lpdbc->p += 2;
                                                                }
                                                        else
                                                                {

        tempFloat = *((STMT *) hstmt)->lpdbc->p;

        ((STMT *) hstmt)->lpdbc->p += 1;
                                                                }
                                                }
                                }

                        switch (((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLLen & 0x00FF)
                                {
                                case 15:
                                        tempFloat /= 1000000000000000.0;
                                        break;
                                case 14:
                                        tempFloat /= 100000000000000.0;
                                        break;
                                case 13:
                                        tempFloat /= 10000000000000.0;
                                        break;
                                case 12:
                                        tempFloat /= 1000000000000.0;
                                        break;
                                case 11:
                                        tempFloat /= 100000000000.0;
                                        break;
                                case 10:
                                        tempFloat /= 10000000000.0;
                                        break;
                                case 9:
                                        tempFloat /= 1000000000.0;
                                        break;
                                case 8:
                                        tempFloat /= 100000000.0;
                                        break;
                                case 7:
                                        tempFloat /= 10000000.0;
                                        break;
                                case 6:
                                        tempFloat /= 1000000.0;
                                        break;
                                case 5:
                                        tempFloat /= 100000.0;
                                        break;
                                case 4:
                                        tempFloat /= 10000.0;
                                        break;
                                case 3:
                                        tempFloat /= 1000.0;
                                        break;
                                case 2:
                                        tempFloat /= 100.0;
                                        break;
                                case 1:
                                        tempFloat /= 10.0;
                                        break;
                                default:;

                                }

                        break;
                case 480:                               /* float */
                        memcpy (&tempFloat, ((STMT *) hstmt)->lpdbc->p,
8);
                        ((STMT *) hstmt)->lpdbc->p += 8;
                        break;
                case 452:
                        tempSmallint = ((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLLen;
                        memcpy (tempStr, (char *) ((STMT *) hstmt)-
>lpdbc->p, __min (tempSmallint, 511));
```

```
                        tempStr[__min (tempSmallint, 511)] = '\0';
                        ((STMT *) hstmt)->lpdbc->p += tempSmallint;
                        break;
                case 448:
                        memcpy (&tempSmallint, ((STMT *) hstmt)->lpdbc-
>p, 2);
                        memcpy (tempStr, (char *) (((STMT *) hstmt)-
>lpdbc->p + 2), __min (tempSmallint, 511));
                        tempStr[__min (tempSmallint, 511)] = '\0';
                        ((STMT *) hstmt)->lpdbc->p += tempSmallint + 2;
                        break;
                case 752:
                        memcpy (&tempInteger, ((STMT *) hstmt)->lpdbc-
>p, 4);
                        tempInteger += 19000000;
                        ((STMT *) hstmt)->lpdbc->p += 4;
                        break;
                case 756:
                        tempSmallint = (short) *(char *) ((STMT *)
hstmt)->lpdbc->p;
                        tempInteger = tempSmallint;
                        ((STMT *) hstmt)->lpdbc->p += 1;
                        break;
                default:
                        *pcbValue = snprintf (rgbValue, cbValueMax,
"%S", "????");
                        ((STMT *) hstmt)->lpdbc->p += ((STMT *) hstmt)-
>lpdbc->DataInfo.InfoVar[icol - 1].SQLLen;
                        break;
                }

        if (fCType == SQL_C_CHAR)
                {
                        ((char *) rgbValue)[0] = '\0';
                        switch (((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLType & 0xFFFE)
                                {
                                case 496:               /* integer */
                                        *pcbValue = snprintf (rgbValue,
cbValueMax, "%10ld", tempInteger);
                                        break;
                                case 500:               /* smallint */
                                        *pcbValue = snprintf (rgbValue,
cbValueMax, "%6hd", tempSmallint);
                                        break;
                                case 484:               /* decimal */
#ifdef WIN32
                                        *pcbValue = snprintf (rgbValue,
cbValueMax, "%15.*f", ((STMT *) hstmt)->lpdbc->DataInfo.InfoVar[icol -
1].SQLLen & 0x00FF, tempFloat);
#else
                                        *pcbValue = sprintf (rgbValue,
"%15.*f", ((STMT *) hstmt)->lpdbc->DataInfo.InfoVar[icol - 1].SQLLen &
0x00FF, tempFloat);
#endif
                                        break;
                                case 480:               /* float */
                                        *pcbValue = snprintf (rgbValue,
cbValueMax, "%15g", tempFloat);
                                        break;
                                case 452:
                                case 448:
                                        *pcbValue = snprintf (rgbValue,
cbValueMax, "%s", tempStr);
                                        break;
                                case 752:
#ifdef WIN32
                                        *pcbValue = snprintf (rgbValue,
cbValueMax, "%04ld-%02d-%02d", tempInteger / 10000, (tempInteger / 100)
% 100, tempInteger % 100);
#else
                                        *pcbValue = sprintf (rgbValue,
"%04ld-%02d-%02d", tempInteger / 10000, (tempInteger / 100) % 100,
tempInteger % 100);
#endif
                                        break;
                                case 756:
                                        *pcbValue = snprintf (rgbValue,
cbValueMax, "%4hd", tempSmallint);
                                        break;
                                default:
                                        *pcbValue = snprintf (rgbValue,
cbValueMax, "%s", "????");
                                        break;
                                }
                        if (*pcbValue < 0)
                                {
                                        fprintf (stderr, " sprintf
failed!\n");
                                }
                        *pcbValue = strlen (rgbValue);

                }
        else if (fCType == SQL_C_LONG || fCType == SQL_C_SLONG)
                {
                        if (pcbValue != NULL)
                                *pcbValue = 4;
                        switch (((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLType & 0xFFFE)
                                {
                                case 496:
                                case 500:
                                case 752:
                                case 756:               /* integer,
smallint, byteint, date */
                                        memcpy (rgbValue, &tempInteger,
4);
                                        break;
                                case 484:
                                case 480:               /* decimal, float
*/
                                        tempInteger = (long) tempFloat;
                                        memcpy (rgbValue, &tempInteger,
4);
                                        break;
```

```
                                default:
                                        tempInteger = 0;
                                        memcpy (rgbValue, &tempInteger,
4);
                                        break;
                                }
                }

        return (SQL_SUCCESS);

}


SQLRETURN SQL_API
SQLFetch (
                        SQLHSTMT hstmt)
{
        Int32 ReturnCode;
        long ActivityCount;




        struct CliSuccessType *successPtr;
        struct CliFailureType *failurePtr;
        struct CliDataInfoType *datainfoPtr;

        ((STMT *) hstmt)->lpdbc->p = NULL;

        ((STMT *) hstmt)->lpdbc->dbcarea.i_sess_id = ((STMT *) hstmt)-
>lpdbc->SessionId;
        ((STMT *) hstmt)->lpdbc->dbcarea.i_req_id = ((STMT *) hstmt)-
>lpdbc->ReqId;
        ((STMT *) hstmt)->lpdbc->dbcarea.change_opts = 'y';



        assert (((STMT *) hstmt)->lpdbc->SessionId > 0);
        assert (((STMT *) hstmt)->lpdbc->ReqId > 0);

        assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
        assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len == sizeof
(struct DBCAREA));

        ((STMT *) hstmt)->lpdbc->dbcarea.func = DBFFET;
        DBCHCL (&ReturnCode, CLICTXT, &((STMT *) hstmt)->lpdbc-
>dbcarea);
        if (((STMT *) hstmt)->lpdbc->Async && ReturnCode == 305)
                {
                        fprintf (stderr, " Fetch still executing? The
FETCH return code was %ld\n", ReturnCode);
                        return (SQL_STILL_EXECUTING);
                }
        if (ReturnCode != 0)
                {
                        fprintf (stderr, " The FETCH return code was
%ld\n", ReturnCode);
                        strncpy (((STMT *) hstmt)->lpdbc->ErrorMessage,
((STMT *) hstmt)->lpdbc->dbcarea.msg_text, ((STMT *) hstmt)->lpdbc-
>dbcarea.msg_len);
                        ((STMT *) hstmt)->lpdbc->ErrorMessage[((STMT *)
hstmt)->lpdbc->dbcarea.msg_len] = '\0';
                        ((STMT *) hstmt)->lpdbc->NativeError =
ReturnCode;
                        return (SQL_ERROR);
                }

       assert (((STMT *) hstmt)->lpdbc->dbcarea.i_sess_id == ((STMT *)
hstmt)->lpdbc->SessionId);
       assert (((STMT *) hstmt)->lpdbc->dbcarea.i_req_id == ((STMT *)
hstmt)->lpdbc->ReqId);


       switch (((STMT *) hstmt)->lpdbc->dbcarea.fet_parcel_flavor)
                {
                case PclSUCCESS:
                        successPtr = (struct CliSuccessType *) ((STMT *)
hstmt)->lpdbc->dbcarea.fet_data_ptr;
                        memcpy (&ActivityCount, successPtr-
>ActivityCount, 4);
                        ((STMT *) hstmt)->lpdbc->RowCount =
ActivityCount;
                        ((STMT *) hstmt)->lpdbc->numResultCols =
successPtr->FieldCount;
                        ((STMT *) hstmt)->lpdbc->busy = 0;
                        ((STMT *) hstmt)->lpdbc->dbcarea.change_opts =
'Y';
                        ((STMT *) hstmt)->lpdbc->dbcarea.wait_for_resp =
'Y';
                        break;
                case PclERROR:
                case PclFAILURE:
                        ((STMT *) hstmt)->lpdbc->busy = 0;
                        ((STMT *) hstmt)->lpdbc->dbcarea.change_opts =
'Y';
                        ((STMT *) hstmt)->lpdbc->dbcarea.wait_for_resp =
'Y';
                        failurePtr = (struct CliFailureType *) ((STMT *)
hstmt)->lpdbc->dbcarea.fet_data_ptr;
                        ((STMT *) hstmt)->lpdbc->NativeError =
failurePtr->Code;
                        strncpy (((STMT *) hstmt)->lpdbc->ErrorMessage,
failurePtr->Msg, (size_t) (failurePtr->Length));
                        ((STMT *) hstmt)->lpdbc-
>ErrorMessage[(failurePtr->Length)] = '\0';

                        fprintf (stderr, " Failure: %d %s\n", ((STMT *)
hstmt)->lpdbc->NativeError, ((STMT *) hstmt)->lpdbc->ErrorMessage);

                        SQLFreeStmt (hstmt, SQL_CLOSE);
                        assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
```

```
                        assert (((STMT *) hstmt)->lpdbc-
>dbcarea.total_len == sizeof (struct DBCAREA));
                        return (SQL_ERROR);
                        /*break; */
                case PclDATAINFO:
                        datainfoPtr = (struct CliDataInfoType *) ((STMT
*) hstmt)->lpdbc->dbcarea.fet_data_ptr;
                        assert (datainfoPtr->FieldCount == ((STMT *)
hstmt)->lpdbc->numResultCols);
                        memcpy (&((STMT *) hstmt)->lpdbc->DataInfo,
datainfoPtr, sizeof (struct CliDataInfoType));
                        assert (((STMT *) hstmt)->lpdbc-
>DataInfo.FieldCount == ((STMT *) hstmt)->lpdbc->numResultCols);
                        assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
                        assert (((STMT *) hstmt)->lpdbc-
>dbcarea.total_len == sizeof (struct DBCAREA));
                        break;
                case PclENDSTATEMENT:
                        ((STMT *) hstmt)->lpdbc->NativeError = 0;
                        ((STMT *) hstmt)->lpdbc->busy = 0;
                        return (SQL_NO_DATA_FOUND);
                        /*break; */
                case PclENDREQUEST:
                        SQLFreeStmt (hstmt, SQL_CLOSE);
                        ((STMT *) hstmt)->lpdbc->NativeError = 0;
                        return (SQL_NO_DATA_FOUND);
                        /*break; */
                case PclRECORD:
                        ((STMT *) hstmt)->lpdbc->NativeError = 0;
                        ((STMT *) hstmt)->lpdbc->p = ((struct
CliRecordType *) ((STMT *) hstmt)->lpdbc->dbcarea.fet_data_ptr)->Body;
                        assert (((STMT *) hstmt)->lpdbc->p != NULL);

                        ((STMT *) hstmt)->lpdbc->p += ((((STMT *)
hstmt)->lpdbc->numResultCols + 7) / 8);      /* Skip Indicator bytes */
                        return (SQL_SUCCESS);
                        /*break; */

                default:
                        fprintf (stderr, " Dont know this parcel
type\n");
                        ((STMT *) hstmt)->lpdbc->NativeError = (-1);
                        return (SQL_ERROR);
                }

        return (SQL_SUCCESS);

}

SQLRETURN SQL_API
SQLMoreResults (
                                SQLHSTMT hstmt)
{
        int rc;
        ((STMT *) hstmt)->lpdbc->dbcarea.i_sess_id = ((STMT *) hstmt)-
>lpdbc->SessionId;
        ((STMT *) hstmt)->lpdbc->dbcarea.i_req_id = ((STMT *) hstmt)-
>lpdbc->ReqId;
        ((STMT *) hstmt)->lpdbc->dbcarea.wait_for_resp = 'Y';
        ((STMT *) hstmt)->lpdbc->dbcarea.change_opts = 'Y';
        ((STMT *) hstmt)->lpdbc->numResultCols = 0;
        ((STMT *) hstmt)->lpdbc->RowCount = 0;

        if ((rc = SQLFetch (hstmt)) != SQL_SUCCESS)
                return ((SQLRETURN) rc);      /* The success parcel */
        assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
        assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len == sizeof
(struct DBCAREA));
        ((STMT *) hstmt)->lpdbc->dbcarea.using_data_len = 0;
        ((STMT *) hstmt)->lpdbc->dbcarea.using_data_ptr = NULL;
        while (((STMT *) hstmt)->lpdbc->dbcarea.fet_parcel_flavor ==
PclRECORD)
                if ((rc = SQLFetch (hstmt)) != SQL_SUCCESS)
                        return ((SQLRETURN) rc);
        if (((STMT *) hstmt)->lpdbc->dbcarea.fet_parcel_flavor ==
PclENDSTATEMENT)
                if ((rc = SQLFetch (hstmt)) != SQL_SUCCESS)
                        return ((SQLRETURN) rc);
        if (((STMT *) hstmt)->lpdbc->dbcarea.fet_parcel_flavor ==
PclENDREQUEST)
                return (SQL_NO_DATA_FOUND);

        if ((rc = SQLFetch (hstmt)) != SQL_SUCCESS)
                {
                        if (rc != SQL_NO_DATA_FOUND)
                                return ((SQLRETURN) rc);      /* Data
Info parcel or End */
                }
        assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
        assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len == sizeof
(struct DBCAREA));
        return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLConnect (
                        SQLHDBC hdbc,
                        SQLCHAR FAR * szDSN,
                        SQLSMALLINT cbDSN,
                        SQLCHAR FAR * szUID,
                        SQLSMALLINT cbUID,
                        SQLCHAR FAR * szAuthStr,
                        SQLSMALLINT cbAuthStr)
{
        SQLHSTMT fakehstmt;

        Int32 ReturnCode;
        static char UserInfo[250];


        assert (((DBC *) hdbc)->dbcarea.total_len == sizeof (struct
DBCAREA));
```

```
        strncpy (((DBC *) hdbc)->dbcarea.eyecatcher, "DBCAREA", 8);

        ((DBC *) hdbc)->dbcarea.req_buf_len = 8192;
        ((DBC *) hdbc)->dbcarea.resp_buf_len = 32700;
        ((DBC *) hdbc)->dbcarea.max_num_sess = 16;
        ((DBC *) hdbc)->dbcarea.resp_mode = 'I';
        ((DBC *) hdbc)->dbcarea.keep_resp = 'N';
        ((DBC *) hdbc)->dbcarea.use_presence_bits = 'N';
        ((DBC *) hdbc)->dbcarea.wait_for_resp = 'Y';
        ((DBC *) hdbc)->dbcarea.loc_mode = 'Y';
        ((DBC *) hdbc)->dbcarea.var_len_req = 'N';
        ((DBC *) hdbc)->dbcarea.var_len_fetch = 'N';
        ((DBC *) hdbc)->dbcarea.save_resp_buf = 'N';
        ((DBC *) hdbc)->dbcarea.two_resp_bufs = 'N';
        ((DBC *) hdbc)->dbcarea.req_proc_opt = 'E';
        ((DBC *) hdbc)->dbcarea.change_opts = 'Y';
        ((DBC *) hdbc)->dbcarea.wait_for_resp = 'Y';
        ((DBC *) hdbc)->dbcarea.ret_time = 'Y';
        ((DBC *) hdbc)->dbcarea.parcel_mode = 'Y';
        ((DBC *) hdbc)->dbcarea.tell_about_crash = 'Y';             /*
Set crash options */
        ((DBC *) hdbc)->dbcarea.wait_across_crash = 'N';     /* Set
crash options */
        ((DBC *) hdbc)->dbcarea.msg_security = 'N';

        strcpy (UserInfo, (char *) szDSN);
        strcat (UserInfo, "/");
        strcat (UserInfo, (char *) szUID);
        strcat (UserInfo, ",");
        strcat (UserInfo, (char *) szAuthStr);
        ((DBC *) hdbc)->dbcarea.logon_ptr = (char *) UserInfo;
        ((DBC *) hdbc)->dbcarea.logon_len = strlen (UserInfo);
        ((DBC *) hdbc)->dbcarea.run_ptr = NULL;
/*      dbcarea.run_ptr = (char *) connect_struct;
   dbcarea.run_len = sizeof (struct CliCONNECTType);
   dbcarea.inter_ptr = CharSet;
   dbcarea.charset_type = CharSetType; */
        ((DBC *) hdbc)->dbcarea.sess_2pc_mode = 'N';
        ((DBC *) hdbc)->dbcarea.using_data_len = 0;
        ((DBC *) hdbc)->dbcarea.using_data_ptr = NULL;


        ((DBC *) hdbc)->dbcarea.func = DBFCON;
        DBCHCL (&ReturnCode, CLICTXT, &((DBC *) hdbc)->dbcarea);
        if (ReturnCode == EM_NETCONN)
                {
                        fprintf (stderr, "Teradata system is down,
waiting to logon...\n");
                        strcpy (((DBC *) hdbc)->ErrorMessage, "Teradata
system is down.");
                        return (-1);
                }
        //while (ReturnCode == EM_NETCONN) {
        //      /* loop until connected or some other error then DBC
down.  */
        //      dbcarea.func = DBFCON;
        //      DBCHCL (&ReturnCode, CLICTXT, dbcarea);
        //   }

        if (ReturnCode != 0)
                {
                        fprintf (stderr, " The logon return code was
%ld\n", ReturnCode);
                        strncpy (((DBC *) hdbc)->ErrorMessage, ((DBC *)
hdbc)->dbcarea.msg_text, ((DBC *) hdbc)->dbcarea.msg_len);
                        ((DBC *) hdbc)->ErrorMessage[((DBC *) hdbc)-
>dbcarea.msg_len] = '\0';
                        ((DBC *) hdbc)->NativeError = ReturnCode;
                }
        ((DBC *) hdbc)->dbcarea.i_sess_id = ((DBC *) hdbc)-
>dbcarea.o_sess_id;
        ((DBC *) hdbc)->dbcarea.i_req_id = ((DBC *) hdbc)-
>dbcarea.o_req_id;
        ((DBC *) hdbc)->SessionId = ((DBC *) hdbc)->dbcarea.o_sess_id;
        ((DBC *) hdbc)->ReqId = ((DBC *) hdbc)->dbcarea.o_req_id;

        fakehstmt = (SQLHSTMT) malloc (sizeof (struct STMT));
        ((STMT *) fakehstmt)->lpdbc = (DBC *) hdbc;
        if (SQLFetch (fakehstmt) == SQL_ERROR)
                return (SQL_ERROR);

        assert (0 == strcmp (((DBC *) hdbc)->dbcarea.eyecatcher,
"DBCAREA"));
        assert (((DBC *) hdbc)->dbcarea.total_len == sizeof (struct
DBCAREA));

        //fprintf(stderr,"Session %d, Req %d logged on\n",((DBC
*)hdbc)->dbcarea.o_sess_id,((DBC *)hdbc)->dbcarea.o_req_id);
        return (SQL_SUCCESS);
}




SQLRETURN SQL_API
SQLExecDirect (
                                        SQLHSTMT hstmt,
                                        SQLCHAR FAR * szSqlStr,
                                        SQLINTEGER cbSqlStr)
{
        Int32 ReturnCode;
        int rc;


        assert (((STMT *) hstmt)->lpdbc->SessionId > 0);
        assert (((STMT *) hstmt)->lpdbc->ReqId >= 0);
        assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
        assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len == sizeof
(struct DBCAREA));

        ((STMT *) hstmt)->lpdbc->dbcarea.i_sess_id = ((STMT *) hstmt)-
>lpdbc->SessionId;
        ((STMT *) hstmt)->lpdbc->dbcarea.i_req_id = ((STMT *) hstmt)-
>lpdbc->ReqId;
```

```
        ((STMT *) hstmt)->lpdbc->dbcarea.change_opts = 'Y';
        if (((STMT *) hstmt)->lpdbc->busy == 0)
                {
                        assert (szSqlStr != NULL);



                        ((STMT *) hstmt)->lpdbc->dbcarea.wait_for_resp =
'Y';
                        ((STMT *) hstmt)->lpdbc->dbcarea.req_ptr = (char
*) szSqlStr;
                        ((STMT *) hstmt)->lpdbc->dbcarea.req_len =
strlen ((char *) szSqlStr);

                        ((STMT *) hstmt)->lpdbc->numResultCols = 0;
                        ((STMT *) hstmt)->lpdbc->RowCount = 0;
                        ((STMT *) hstmt)->lpdbc->dbcarea.token = (Int32)
hstmt;

                        ((STMT *) hstmt)->lpdbc->dbcarea.func = DBFIRQ;
                        DBCHCL (&ReturnCode, CLICTXT, &((STMT *) hstmt)-
>lpdbc->dbcarea);
                        if (ReturnCode != 0)
                                {
                                        fprintf (stderr, " The IRQ return
code was %ld\n", ReturnCode);
                                        strncpy (((STMT *) hstmt)->lpdbc-
>ErrorMessage, ((STMT *) hstmt)->lpdbc->dbcarea.msg_text, ((STMT *)
hstmt)->lpdbc->dbcarea.msg_len);
                                        ((STMT *) hstmt)->lpdbc-
>ErrorMessage[((STMT *) hstmt)->lpdbc->dbcarea.msg_len] = '\0';
                                        ((STMT *) hstmt)->lpdbc-
>NativeError = ReturnCode;
                                        return (SQL_ERROR);
                                }
                        ((STMT *) hstmt)->lpdbc->busy = 1;
                        //fprintf(stderr,"SQL request running on Session
%d, Req %d\n",((STMT*)hstmt)->lpdbc->dbcarea.o_sess_id,((STMT*)hstmt)-
>lpdbc->dbcarea.o_req_id);
                        if (((STMT *) hstmt)->lpdbc->Async ==
SQL_ASYNC_ENABLE_ON)
                                return SQL_STILL_EXECUTING;
                }

        //assert(((STMT *)hstmt)->lpdbc->SessionId ==
dbcarea.o_sess_id);
        ((STMT *) hstmt)->lpdbc->SessionId = ((STMT *) hstmt)->lpdbc-
>dbcarea.o_sess_id;
        ((STMT *) hstmt)->lpdbc->ReqId = ((STMT *) hstmt)->lpdbc-
>dbcarea.o_req_id;
        assert (((STMT *) hstmt)->lpdbc->SessionId > 0);
        assert (((STMT *) hstmt)->lpdbc->ReqId >= 0);
        assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
        assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len == sizeof
(struct DBCAREA));
        ((STMT *) hstmt)->lpdbc->dbcarea.i_sess_id = ((STMT *) hstmt)-
>lpdbc->SessionId;
        ((STMT *) hstmt)->lpdbc->dbcarea.i_req_id = ((STMT *) hstmt)-
>lpdbc->ReqId;

        rc = SQLFetch (hstmt);
        if (rc != SQL_SUCCESS)
                return ((SQLRETURN) rc);      /* The success parcel */
        assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
        assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len == sizeof
(struct DBCAREA));
        ((STMT *) hstmt)->lpdbc->dbcarea.using_data_len = 0;
        ((STMT *) hstmt)->lpdbc->dbcarea.using_data_ptr = NULL;
        ((STMT *) hstmt)->lpdbc->dbcarea.wait_for_resp = 'Y';
        ((STMT *) hstmt)->lpdbc->dbcarea.change_opts = 'Y';
        ((STMT *) hstmt)->lpdbc->busy = 0;

        rc = SQLFetch (hstmt);
        if (rc != SQL_SUCCESS)
                {
                        if (rc == SQL_NO_DATA_FOUND)
                                return SQL_SUCCESS;
                        return ((SQLRETURN) rc);      /* Data Info
parcel or End */
                }
        assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
        assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len == sizeof
(struct DBCAREA));
        return (SQL_SUCCESS);

}

SQLRETURN SQL_API
SQLDisconnect (
                                SQLHDBC hdbc)
{
        Int32 ReturnCode;

        assert (((DBC *) hdbc)->SessionId > 0);
        assert (((DBC *) hdbc)->ReqId >= 0);
        assert (0 == strcmp (((DBC *) hdbc)->dbcarea.eyecatcher,
"DBCAREA"));
        assert (((DBC *) hdbc)->dbcarea.total_len == sizeof (struct
DBCAREA));
        ((DBC *) hdbc)->dbcarea.i_sess_id = ((DBC *) hdbc)->SessionId;
        ((DBC *) hdbc)->dbcarea.i_req_id = ((DBC *) hdbc)->ReqId;
        ((DBC *) hdbc)->dbcarea.using_data_len = 0;
        ((DBC *) hdbc)->dbcarea.using_data_ptr = NULL;
        ((DBC *) hdbc)->dbcarea.func = DBFDSC;
        DBCHCL (&ReturnCode, CLICTXT, &((DBC *) hdbc)->dbcarea);
        if (ReturnCode != 0)
                {
                        fprintf (stderr, " The LOGOFF return code was
%ld\n", ReturnCode);
                        strncpy (((DBC *) hdbc)->ErrorMessage, ((DBC *)
hdbc)->dbcarea.msg_text, ((DBC *) hdbc)->dbcarea.msg_len);
```

```
                        ((DBC *) hdbc)->ErrorMessage[((DBC *) hdbc)-
>dbcarea.msg_len] = '\0';
                        ((DBC *) hdbc)->NativeError = ReturnCode;
                        return (SQL_ERROR);
                }

/*  DBCHCLN (&ReturnCode, CLICTXT);
   if (ReturnCode!=0) {
   fprintf(stderr," The HCLN return code was %ld\n",ReturnCode);
   strncpy(((DBC *)hdbc)-
>ErrorMessage,dbcarea.msg_text,dbcarea.msg_len);
   ((DBC *)hdbc)->ErrorMessage[dbcarea.msg_len]='\0';
   ((DBC *)hdbc)->NativeError= ReturnCode;
   return(SQL_ERROR);
   }       */

        return (SQL_SUCCESS);

}

SQLRETURN SQL_API
SQLSetUsingParcel (
                                        SQLHSTMT hstmt,
                                        char *uptr,
                                        long ulen)
{
        ((STMT *) hstmt)->lpdbc->dbcarea.using_data_ptr = uptr;
        ((STMT *) hstmt)->lpdbc->dbcarea.using_data_len = ulen;
        return (SQL_SUCCESS);

}

SQLRETURN SQL_API
SQLWaitForAny (
                                SQLHENV henv,
                                SQLHSTMT * lphstmt,
                                int *stmtno)
{
        Int32 ReturnCode;
        Int32 sessionid;
        Int32 reqtoken;
        STMT *lpstmt;
        int rc;
        sessionid = 0;
        reqtoken = 0;
        *lphstmt = SQL_NULL_HSTMT;
        *stmtno = -1;

        DBCHWAT (&ReturnCode, CLICTXT, &sessionid, &reqtoken);
        if (ReturnCode == NOACTIVE)
                {
                        fprintf (stderr, " DBCHWAT says nothing was
active %ld, %ld\n", sessionid, reqtoken);
                        return SQL_NO_DATA_FOUND;
                }
        if (ReturnCode != 0)
                {
                        fprintf (stderr, " DBCHWAT returned %d\n",
ReturnCode);
                        return SQL_ERROR;
                }

        lpstmt = (STMT *) reqtoken;
        if (lpstmt->lpdbc->SessionId != sessionid)
                fprintf (stderr, " DBCHWAT returned the wrong
session?\n");
        *lphstmt = (SQLHSTMT) lpstmt;
        *stmtno = lpstmt->hstmt_num;

        if (lpstmt->lpdbc->busy)
                {
                        //fprintf(stderr," DBCHWAT returned a busy
session completion on %d\n",sessionid);
                        lpstmt->lpdbc->dbcarea.wait_for_resp = 'Y';
                        lpstmt->lpdbc->dbcarea.change_opts = 'Y';
                        rc = SQLExecDirect ((SQLHSTMT) lpstmt, (SQLCHAR
*) NULL, 0);
                        if (rc != SQL_SUCCESS && rc !=
SQL_SUCCESS_WITH_INFO)
                                {
                                        //fprintf (stderr, " Completion
of request failed! rc = %d\n", rc);
                                        if (rc == SQL_NO_DATA_FOUND)
                                                rc = SQL_SUCCESS;
                                        return (SQLRETURN) rc;
                                }
                }

        //fprintf(stderr," ***DBCHWAT returned an idle session
%d\n",sessionid);
        return SQL_SUCCESS;


}
```